日 本 国 特 許 庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出 願 年 月 日
Date of Application: ２００３年 １月 ９日

出 願 番 号
Application Number: 特願２００３－００３４２８

[ST. 10/C]: [ＪＰ２００３－００３４２８]

出 願 人
Applicant(s): 株式会社東芝

２００３年 ７月 ８日

特許庁長官
Commissioner,
Japan Patent Office 太田信一郎

出証番号 出証特２００３－３０５４０１２

【書類名】　特許願

【整理番号】　13890301

【提出日】　平成15年 1月 9日

【あて先】　特許庁長官殿

【国際特許分類】　H06L　9/10

【発明の名称】　プロセッサ、実行タスク決定装置及び演算処理方法

【請求項の数】　17

【発明者】

【住所又は居所】　神奈川県川崎市幸区小向東芝町１番地　株式会社東芝研究開発センター内

【氏名】　鏑　木　　智

【発明者】

【住所又は居所】　神奈川県川崎市幸区小向東芝町１番地　株式会社東芝研究開発センター内

【氏名】　宮　本　幸　昌

【発明者】

【住所又は居所】　神奈川県川崎市幸区小向東芝町１番地　株式会社東芝研究開発センター内

【氏名】　菅　野　伸　一

【発明者】

【住所又は居所】　神奈川県川崎市幸区小向東芝町１番地　株式会社東芝研究開発センター内

【氏名】　樽　家　昌　也

【発明者】

【住所又は居所】　神奈川県川崎市幸区小向東芝町１番地　株式会社東芝研究開発センター内

【氏名】　大根田　　拓

【特許出願人】
　【識別番号】　000003078
　【住所又は居所】　東京都港区芝浦一丁目１番１号
　【氏名又は名称】　株式会社　東　芝
【代理人】
　【識別番号】　100075812
　【弁理士】
　【氏名又は名称】　吉　武　賢　次
【選任した代理人】
　【識別番号】　100088889
　【弁理士】
　【氏名又は名称】　橘　谷　英　俊
【選任した代理人】
　【識別番号】　100082991
　【弁理士】
　【氏名又は名称】　佐　藤　泰　和
【選任した代理人】
　【識別番号】　100096921
　【弁理士】
　【氏名又は名称】　吉　元　弘
【選任した代理人】
　【識別番号】　100103263
　【弁理士】
　【氏名又は名称】　川　崎　康
【手数料の表示】
　【予納台帳番号】　087654
　【納付金額】　21,000円
【提出物件の目録】
　【物件名】　明細書　1

【物件名】　図面　1

【物件名】　要約書　1

【プルーフの要否】　要

【書類名】　明細書

【発明の名称】　プロセッサ、実行タスク決定装置及び演算処理方法

【特許請求の範囲】

【請求項１】

複数のタスクを時分割で処理するプロセッサにおいて、

予め定めたタスクに対して、該タスクの処理で使用されるデータを記憶する１以上の第１のFIFO（First In First Out）型記憶手段と、

前記第１のFIFO型記憶手段から取得したデータを使用して前記タスクの処理を行った後の処理結果を記憶するとともに、この記憶した処理結果を使用する他のタスクがある場合には該他のタスクの処理で使用されるデータを記憶する１以上の第２のFIFO型記憶手段と、

前記第１のFIFO型記憶手段が前記タスクの処理に使用するデータを保持しているか否か、および前記タスクの処理結果を記憶する空き領域を前記第２のFIFO型記憶手段が有しているか否かを判断するタスク判断手段と、

前記タスク判断手段の判断結果に基づいて、前記複数のタスクのうちから次に起動すべきタスクを決定する実行タスク決定手段と、を備えることを特徴とするプロセッサ。

【請求項２】

前記複数のタスクのそれぞれに優先順位を設定する優先順位設定手段を備え、

前記実行タスク決定手段は、前記優先順位設定手段により設定された優先順位に基づいて、前記起動すべきタスクを決定することを特徴とする請求項１に記載のプロセッサ。

【請求項３】

前記複数のタスクそれぞれの起動頻度を計測する起動頻度計測手段を備え、

前記優先順位設定手段は、前記起動頻度計測手段で計測された起動頻度に基づいて、前記優先順位を設定することを特徴とする請求項２に記載のプロセッサ。

【請求項４】

前記優先順位設定手段は、前記起動頻度がより高いタスクほど優先順位を高く設定することを特徴とする請求項３に記載のプロセッサ。

【請求項５】

前記実行タスク決定手段は、第１のタスクの起動中に、該第１のタスクよりも優先順位の高い第２のタスクに使用されるデータを前記第１のFIFO型記憶手段が保持し、かつ前記第２のタスクの処理結果を格納すべき空き領域を前記第２のFIFO型記憶手段が有する場合、前記第１のタスクの起動を中断して、前記第２のタスクを起動することを特徴とする請求項２に記載のプロセッサ。

【請求項６】

個々のタスクに固有の情報をすべて含む１以上のレジスタからなるレジスタセットを、時分割で実行可能なタスクの総数以上設けたレジスタセット群と、

前記実行タスク決定手段により決定したタスクが利用する前記レジスタセットを選択するレジスタセット選択手段と、

前記レジスタセット選択手段で選択したレジスタセットを利用して、前記実行タスク決定手段により決定したタスクの処理を行う演算処理手段と、を備えることを特徴とする請求項１に記載のプロセッサ。

【請求項７】

タスクと、該タスクが利用する前記レジスタセットと、の対応関係を記憶する対応関係記憶手段を備え、

前記レジスタセット選択手段は、前記対応関係記憶手段に記憶した対応関係に基づいて、前記実行タスク決定手段によって決定したタスクが利用するレジスタセットを選択することを特徴とする請求項６に記載のプロセッサ。

【請求項８】

個々のタスクに固有の情報をすべて含む１以上のレジスタからなるレジスタセットを、時分割で実行可能なタスクの総数に満たない数だけ設けたレジスタセット群と、

前記実行タスク決定手段により決定したタスクが利用する前記レジスタセットを選択するレジスタセット選択手段と、

前記レジスタセット選択手段で選択したレジスタセットを利用して、前記実行タスク決定手段により決定したタスクの処理を行う演算処理手段と、

前記レジスタセット群に含まれる任意のレジスタセットの内容を記憶可能な外

部レジスタセット記憶手段と、

前記タスクが利用するレジスタセットの内容を、前記外部レジスタセット記憶手段に記憶している場合には、

前記レジスタセット群に含まれる、中断中か中断すべきタスクが利用するレジスタセットの内容を前記外部レジスタセット記憶手段に転送するとともに、前記外部レジスタセット記憶手段に記憶した前記タスクが利用するレジスタセットの内容を、前記レジスタセット群のうちの前記外部レジスタセット記憶手段に転送したレジスタセットに対応するレジスタセットに転送する記憶制御手段と、を備えることを特徴とする請求項１に記載のプロセッサ。

【請求項９】

前記実行タスク決定手段により決定されたタスクが利用する前記レジスタセットが前記レジスタセット群の中に存在するか否かを判定するヒット判定手段を備え、

前記ヒット判定手段が、前記タスクが利用するレジスタセットが前記レジスタセット群に存在すると判定した場合には、

前記レジスタセット選択手段は、前記レジスタセット群のうちの前記レジスタセットを選択し、

他方、存在しないと判断した場合には、

前記記憶制御手段が前記外部レジスタセット記憶手段に記憶している前記タスクが利用するレジスタセットの内容を、前記レジスタセット群のうちの転送先に対応するレジスタセットに転送した後に、前記レジスタセット選択手段はこの転送後のレジスタセットを選択することを特徴とする請求項８に記載のプロセッサ。

【請求項１０】

前記ヒット判定手段により存在しないと判定したときに、前記外部レジスタセット記憶手段に待避させるべき前記レジスタセット群のレジスタセットを選択する待避レジスタセット決定手段を備え、

前記記憶制御手段は、前記待避レジスタセット決定手段により選択されたレジスタセットの内容を前記外部レジスタセット記憶手段に転送することを特徴とす

る請求項９に記載のプロセッサ。

【請求項１１】

前記実行タスク決定手段は、前記タスク判断手段による判断結果を考慮に入れるとともに、前記記憶制御手段が前記レジスタセット群と前記外部レジスタセット記憶手段との間でレジスタセットの入れ替え処理を行っている最中にも、他のタスクの処理ができるように該入れ替え処理の影響を受けずに実行可能なタスクを決定することを特徴とする請求項８に記載のプロセッサ。

【請求項１２】

前記記憶制御手段は、前記外部レジスタセット記憶手段の記憶内容を更新する必要がある場合、前記レジスタセット群の変更のあったレジスタの値のみ、前記外部レジスタセット記憶手段に転送することを特徴とする請求項８に記載のプロセッサ。

【請求項１３】

前記記憶制御手段は、前記外部レジスタセット記憶手段から前記レジスタセット群にデータを転送する必要がある場合には、他のタスクの実行によって前記レジスタセット群のレジスタの値が書き換えられてしまったレジスタの内容のみ、前記外部レジスタセット記憶手段から前記レジスタセット群に転送することを特徴とする請求項８のいずれかに記載のプロセッサ。

【請求項１４】

前記記憶制御手段は、前記外部レジスタセット記憶手段に記憶されているレジスタセットと同じレジスタセットが前記レジスタセット群に存在しない場合に限り、該レジスタセットの内容を前記外部レジスタセット記憶手段に転送することを特徴とする請求項８に記載のプロセッサ。

【請求項１５】

前記外部レジスタセット記憶手段は、変更のあったレジスタの値及び該レジスタの識別情報だけを記憶することを特徴とする請求項８に記載のプロセッサ。

【請求項１６】

複数のタスクを時分割で処理するプロセッサの、前記複数のタスクの中から次に起動すべきタスクを決定する実行タスク決定装置において、

前記複数のタスクのそれぞれについて、該タスクの処理の入力データを記憶するための１以上の第１のFIFO型記憶手段に、前記タスクが処理可能なデータが記憶されているか否か、および前記タスクの処理結果を記憶するための１以上の第２のFIFO型記憶手段に、該処理結果のデータを記憶できる空き領域があるか否かを判断し、この判断に基づいて前記複数のタスクの中から次に起動すべきタスクを決定する実行タスク決定手段を備えることを特徴とする実行タスク決定装置。

【請求項１７】

複数のタスクを時分割で処理する演算処理方法において、

前記複数のタスクのそれぞれについて、該タスクの処理の入力データを記憶するための１以上の第１のFIFO型記憶手段に、前記タスクが処理可能なデータが記憶されているか否か、および前記タスクの処理結果を記憶するための１以上の第２のFIFO型記憶手段に、該処理結果のデータを記憶できる空き領域があるか否かを判断し、この判断に基づいて前記複数のタスクの中から次に起動すべきタスクを決定することを特徴とする演算処理方法。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は、複数のタスクを時分割で実行するプロセッサ及び実行タスク決定装置に関する。

【０００２】

【従来の技術】

プロセッサが時分割で複数のタスクを実行する場合、オペレーティングシステムなどのソフトウェアを用いてタスク切り替えを行うのが一般的である。

【０００３】

オペレーティングシステムは、一定時間ごとに、各タスクが実行可能になったかどうかを検出する。実行可能の可否は、たとえば処理対象となるデータが準備できているかで判断できる。

【０００４】

このような判断処理をオペレーティングシステムが行っている間は、タスクの

実行を中断しなければならない。タスクの実行を中断する際には、汎用レジスタ、スタックポインタ（SP）及びプログラムカウンタ（PC）などの内容がプロセッサ外部の主記憶メモリなどに退避される。退避された各種の情報は タスクの識別情報（タスクID）とともにオペレーティングシステムが管理する。

【０００５】

オペレーティングシステムはスケジューラを持っており、このスケジューラにより、たとえばラウンドロビンなどの方式に従って、実行可能なタスクの中から実際に実行するタスクを決定する。

【０００６】

オペレーティングシステムは、スケジューラにより実行が決定されたタスクについての、主記憶メモリ等に退避された各種情報を取得し、汎用レジスタ、スタックポインタ（SP）及びプログラムカウンタ（PC）などの内容を退避先のプロセッサ外の主記憶装置などから各レジスタに復元し、その後タスクの実行を開始する。

【０００７】

ハードウェアによる複数のデータ処理ユニットを用いてリアルタイムデータの並列処理を実現する方法も提案されている（たとえば特許文献１を参照）。しかしながら並列処理を行うためにデータ処理コアを複数設け、さらにそれらの調停機構を設けるなど、構成自体が複雑となる問題点がある。

【０００８】

【特許文献１】

特表２０００－５０９５２８公報

【０００９】

【発明が解決しようとする課題】

オペレーティングシステムなどのソフトウェアを用いたタスク切り替えでは、オペレーティングシステム自体の動作によるオーバーヘッドがあるため、本来のタスク処理に必要な性能よりも高い性能のプロセッサを用意する必要があった。

【００１０】

また、オペレーティングシステムによるタスク切り替えでは、タスク切り替え

に時間がかかり、リアルタイム性のあるシステムの構築が困難であった。

【００１１】

さらにオペレーティングシステムは、タスクが実行可能かどうかの判断を一定時間ごとにしか行わないため、タスクが実行可能になってからオペレーティングシステムに実行可能と判断され、実際にタスクが実行されるまでに時間がかかることがあり、応答性が悪かった。この応答性の悪さがリアルタイム性のあるシステムの構築をさらに困難にしていた。

【００１２】

また、処理対象となるデータの準備が出来たかどうかを条件としてタスク起動可能を判定する場合、処理結果を格納する領域にデータを格納出来ない場合があり、そのような場合はデータを失うことになる。

【００１３】

一方、多くをハードウェア化しリアルタイム処理を実現しようとすると、従来は構成が複雑かつコスト高になりやすい傾向があった。

【００１４】

本発明は、このような点に鑑みてなされたものであり、その目的は、複数のタスクをリアルタイムに効率よく実行できるプロセッサおよび実行タスク決定装置を提供することにある。

【００１５】

【課題を解決するための手段】

上述した課題を解決するために、本発明は、複数のタスクを時分割で処理するプロセッサにおいて、予め定めたタスクに対して、該タスクの処理で使用されるデータを記憶する１以上の第１のFIFO（First In First Out）型記憶手段と、前記第１のFIFO型記憶手段から取得したデータを使用して前記タスクの処理を行った後の処理結果を記憶するとともに、この記憶した処理結果を使用する他のタスクがある場合には該他のタスクの処理で使用されるデータを記憶する１以上の第２のFIFO型記憶手段と、前記第１のFIFO型記憶手段が前記タスクの処理に使用するデータを保持しているか否か、および前記タスクの処理結果を記憶する空き領域を前記第２のFIFO型記憶手段が有しているか否かを判断するタスク判断手段と

、前記タスク判断手段の判断結果に基づいて、前記複数のタスクのうちから次に起動すべきタスクを決定する実行タスク決定手段と、を備える。

【００１６】

【発明の実施の形態】

以下、本発明に係るプロセッサ及び実行タスク決定装置について、図面を参照しながら具体的に説明する。

【００１７】

（本発明の原理）

複数のタスクを時分割でリアルタイムに実行したいという要求が高まっている。例えば、図１はソフトウェア無線機の処理内容を示す図である。図１のソフトウェア無線機では、処理Ａ～処理Ｄを連続的に実行する。例えば、処理Ｂは、処理Ａの出力を入力として所定の演算処理を行った結果を処理Ｃに渡す。

【００１８】

図１において、各処理での出力データ数とその次の処理を起動するのに必要なデータ数とは必ずしも一致しない。また、各処理の処理量も処理内容によってさまざまである。さらに、処理間でデータ受け渡しのタイミングが異なるため、各処理の段間で同期を取る必要がある。

【００１９】

図１の処理を効率よく実現するために、各処理を１つ以上のタスクとして生成することが多い。各タスクは、ソフトウェアで実現される場合とハードウェアで実現される場合がある。

【００２０】

本発明では、タスク同士の厳密な同期を取らなくても済むように、タスク間にFIFO（First In First Out）を挿入する。FIFOとは一般にはデータの格納方法を指すが、以下に示す実施形態では格納したデータが過去に格納したものから順に取り出される方式、つまりFIFO型の記憶装置を指している。一番新しく格納されたデータがもっとも最後に取り出されるもので、たとえばキューと呼ばれるデータ構造はこの方式の記憶装置である。この場合のプロセッサの概略構成は図２のようになる。

【００２１】

図２において、FIFO1はタスク１への入力データを出力し、FIFO2はタスク１からの出力データを受け取るとともに、タスク２への入力データを出力する。

【００２２】

図２では、各タスクの入力側及び出力側に１つずつFIFOを設けているが、複数のFIFOを設けてもよい。

【００２３】

ソフトウェア無線機を図２の構成で実現することも可能であるが、各タスク同士で同期を取るための処理量が多いことから、FIFOを利用したとしても、ソフトウェアで同期を取るのは、プロセッサのオーバーヘッドが大きくなり、実現が困難である。

【００２４】

ここで、各タスクの起動条件に着目すると、各タスクは入力データが存在しないときには起動する必要はない。また、各タスクの出力側のFIFOに各タスクの処理結果を格納する空き領域が存在しない場合には、各タスクの処理結果を失うおそれがあり、この場合もタスクを起動する必要はない。

【００２５】

そこで、以下に説明する本発明の各実施形態は、FIFO内のデータの格納状態を監視しておき、各タスクへの入力データが存在し、かつ出力側のFIFOに空き領域が存在する場合のみ、タスクを起動している。

【００２６】

（第１の実施形態）

図３は本発明に係るプロセッサの第１の実施形態の概略構成を示すブロック図である。図３のプロセッサは、複数のタスクそれぞれが起動可能か否かを判断するタスク起動部１１と、タスク起動部１１の判断結果に基づいて起動すべきタスクを決定する実行タスク決定部１２と、実行タスク決定部１２で決定されたタスクを実行するプロセッサコア１３とを備えている。

【００２７】

タスク起動部１１には、複数のFIFO１０が接続されている。各FIFO１０は、図

２のFIFO１０に対応するものであり、タスク起動部１１は各FIFO１０間でデータの受け渡しを行う。より具体的には、タスク起動部１１は、タスクに入力されるべきFIFO１０の出力データを保持するとともに、タスクの処理結果を格納すべき他のFIFO１０に空き領域が存在するか否かを確認する。そして、タスクに入力されるべきFIFO１０の出力データが存在し、かつタスクの処理結果を格納すべき他のFIFO１０に空き領域が存在する場合に限り、タスクを起動すべきと判断する。

【００２８】

実行タスク決定部１２は、起動可能なタスクが複数存在する場合には、その中からいずれか１つのタスクを選択する。例えば、各タスクを識別するためのタスクIDが最小のタスクを実行する。

【００２９】

プロセッサコア１３は、例えば、命令取得部、命令解釈部、命令実行部、命令実行結果を格納する命令実行結果格納部、及び命令の実行状態を格納する命令実行状態格納部を有する。

【００３０】

このように、第１の実施形態では、タスクに入力されるべきデータをFIFO１０が保持し、かつタスクの実行結果を格納すべき他のFIFO１０に空き領域が存在するか否かをタスク起動部１１で判断し、その判断結果に基づいて実行タスク決定部１２が起動すべきタスクを決定するようにしたため、オペレーティングシステムなしでタスクのスケジューリングを行うことができ、オペレーティングシステムのオーバーヘッドの影響を受けなくなり、タスクのリアルタイム処理が可能になる。

【００３１】

（第２の実施形態）

第２の実施形態は、複数のタスクに優先順位をつけて、優先順位に従ってタスクの実行を行うものである。

【００３２】

図４は本発明に係るプロセッサの第２の実施形態の概略構成を示すブロック図である。図４のプロセッサは、図３の構成に加えて、複数のタスクそれぞれのタ

スクIDと優先順位を供給するタスクID・優先順位供給部１４を備えている。

【００３３】

図５は実行タスク決定部１２とタスクID・優先順位供給部１４の詳細構成を示すブロック図である。図示のように、タスクID・優先順位供給部１４は、各タスクのタスクIDと優先順位との対応関係を記憶している。実行タスク決定部１２は、タスク起動部１１から通知された実行可能なタスクの一覧を登録した実行可能タスクIDリスト１５と、起動すべきタスクを決定するプライオリティ・エンコーダ１６とを有する。

【００３４】

プライオリティ・エンコーダ１６は、タスクIDリストに登録されているタスクの優先順位をタスクID・優先順位供給部１４から取得し、例えば優先順位が最も高いタスクを起動タスクとして決定する。

【００３５】

タスクID・優先順位供給部１４が供給する優先順位の設定方法については特に制限はないが、例えば、新たなタスクを生成するときに、そのタスクの優先順位を設定してもよいし、事後的に優先順位を変更できるようにしてもよい。

【００３６】

図６は図５の実行タスク決定部１２の処理手順を示すフローチャートである。まず、実行可能タスクIDリスト１５に登録されているタスクのタスクIDを参照し（ステップＳ１）、そのタスクIDに対応する優先順位をタスクID・優先順位供給部１４から取得する（ステップＳ２）。

【００３７】

次に、ステップＳ２で取得した優先順位が、タスクの起動候補の優先順位より高いか否かを判定し（ステップＳ３）、高ければ、ステップＳ１で参照したタスクIDをもつタスクを起動候補として設定する（ステップＳ４）。

【００３８】

ステップＳ４の処理が終了した場合、またはステップＳ３で高くないと判定された場合には、まだ優先順位を比較していないタスクが実行可能タスクIDリスト１５に残っているか否かを判定し（ステップＳ５）、残っている場合にはステッ

プＳ１以降の処理を繰返し、残っていない場合には処理を終了する。

【００３９】

このように、第２の実施形態では、タスクの優先順位を予め定めておくため、複数のタスクが起動候補として挙げられた場合に、優先順位の高いタスクを優先的に実行でき、処理の効率化が図れる。また、起動すべきタスクの選択を迅速に行うこともできる。

【００４０】

（第３の実施形態）

第３の実施形態は、動的に優先順位を変更できるようにしたものである。

【００４１】

図７は本発明に係るプロセッサの第３の実施形態の概略構成を示すブロック図である。図７のプロセッサは、図４の構成に加えて、タスクの優先順位を動的に変更する優先順位変更部３０を備えている。

【００４２】

優先順位変更部３０は、タスクごとの実行時刻を計測する時刻計測部１７と、各タスクのタスクIDと過去の起動時刻との対応関係を登録した起動時刻テーブル１８と、各タスクの起動間隔の平均を計算する平均時間間隔計算部１９と、平均時間間隔計算部１９での計算結果に基づいて各タスクの優先順位を決定する優先順位決定部２０とを有する。

【００４３】

平均時間間隔計算部１９の計算結果が小さいほど、より頻繁に起動されるタスクであることを示している。このため、優先順位決定部２０は、例えば、平均時間間隔計算部１９の計算結果が小さいタスクほど、優先順位を高くする。

【００４４】

より具体的には、平均時間間隔計算部１９は各FIF0１０へのデータ入力間隔を算出し、その間隔の短いFIF0１０に対応するタスクの優先順位が高くなるように、優先順位決定部２０で優先順位を決定する。あるいは、平均時間間隔計算部１９はプロセッサコア１３から通知される各タスクの実行間隔を算出し、その間隔の短いタスクの優先順位が高くなるように、優先順位決定部２０で優先順位を決

定してもよい。あるいは、平均時間間隔計算部１９は各タスクにデータを供給する各FIF0１０の空き領域を監視し、空き領域の少ないFIF0１０に対応するタスクの優先順位が高くなるように、優先順位決定部２０で優先順位を決定してもよい。あるいは、平均時間間隔計算部１９は各タスクから各FIF0１０へのデータ出力間隔を算出し、その間隔の短いFIF0１０に対応するタスクの優先順位が高くなるように、優先順位決定部２０で優先順位を決定してもよい。

【００４５】

このように、第３の実施形態では、各タスクの優先順位を動的に変更できるようにしたため、過去の実行履歴を参考にしてタスクのスケジューリングを行うことができる。

【００４６】

（第４の実施形態）

第４の実施形態は、実行中のタスクよりも優先順位の高いタスクの起動準備が整うと、実行中のタスクを中断して、優先順位の高いタスクを起動するものである。

【００４７】

第４の実施形態のブロック構成は、図４や図７と同様である。図８は本発明に係るプロセッサの第４の実施形態の動作を説明するタイムチャートである。図８は３つのタスクＡ，Ｂ，Ｃが存在し、タスクＡの優先順位が最も高く、次にタスクＢの優先順位が高く、タスクＣの優先順位が最も低い例を示している。

【００４８】

まず、時刻ｔ０でタスクＣが起動したとする。その後、時刻ｔ１でタスクＢの起動準備が整ったとする。これにより、実行タスク決定部１２はタスクＣの実行を中断して、代わりにタスクＢの起動を開始する。

【００４９】

その後、時刻ｔ２でタスクＡの起動準備が整ったとする。これにより、実行タスク決定部１２はタスクＢの実行を中断して、代わりにタスクＡの起動を開始する。

【００５０】

その後、時刻ｔ３でタスクＡの実行が終了したとすると、次に優先順位の高いタスクＢの実行を再開し、時刻ｔ４でタスクＢの実行が終了したとすると、次に一番優先順位の低いタスクＣの実行を開始する。

【００５１】

このように、第４の実施形態では、実行中のタスクよりも優先順位の高いタスクの起動準備が整うと、実行中のタスクの処理を中断して、優先順位の高いタスクの実行を開始するようにしたため、重要なタスクを常に優先的に処理でき、プロセッサ全体の処理能力の向上が図れる。

【００５２】

（第５の実施形態）

以下に説明する第５～第１１の実施形態は、実行中のタスクを迅速に切り替える仕組みに関する。

【００５３】

図９は本発明に係るプロセッサの第５の実施形態の概略構成を示すブロック図である。図９のプロセッサは、図４と同様のタスク起動部１１、実行タスク決定部１２及びタスクID・優先順位供給部１４を備えており、その他に、複数のレジスタセットからなるレジスタセット群２１と、レジスタセット群２１の中からいずれか１つのレジスタセットを選択するレジスタセット選択部２２と、選択されたレジスタセットを用いて演算処理を行う演算部２３と、を備えている。これらレジスタセット、レジスタセット選択部２２及び演算部２３でプロセッサコア１３を構成している。

【００５４】

レジスタセット選択部２２の内部には、レジスタセット群２１の中から１つのレジスタセットを選択するデコーダ２２ａが設けられている。このデコーダ２２ａは、実行タスク決定部１２からのタスクIDに対応するレジスタセットの選択信号を出力する。

【００５５】

各レジスタセットは、個々のタスクに固有の情報をすべて含む１以上のレジスタからなる。レジスタセットを構成する各レジスタの種類は、プロセッサのアー

キテクチャに依存し、例えば、プログラムカウンタ（PC）、スタックポインタ（SP）、または汎用レジスタ（R0,R1,…）などである。本明細書では、１つのレジスタセット内のレジスタの総数をｒ（ｒは１以上の整数）とする。

【００５６】

レジスタセットの総数は、時分割で実行可能なタスクの数ｎ以上のｍ個（ｍは整数）である。各レジスタセットは、固有の識別番号（レジスタセットID）で識別される。

【００５７】

実行中のタスクは、１つのレジスタセットを用いる。タスクごとに使用するレジスタの種類が異なる場合には、各タスクごとに専用のレジスタセットを設けてもよい。プロセッサの設計時に、レジスタセットを構成するレジスタの種類を決めてもよいし、プログラム命令により事後的にレジスタセットを構成するレジスタの種類を変更してもよい。

【００５８】

タスクごとに使用するレジスタの種類が異なる場合には、必要に応じて、図１０に示すようなタスクIDとレジスタセットIDとの対応関係を登録したレジスタセットIDタスクID対応部２４を設け、このテーブルを参照して、各タスクに対応するレジスタセットを決定してもよい。

【００５９】

レジスタセットIDタスクID対応部２４は、より詳細には、図１１のようなテーブルで構成される。図１１のテーブルは、プロセッサの設計時に作成してその後の変更ができないようにしてもよいし、プロセッサの起動後に何らかの命令によりテーブルの内容を変更できるようにしてもよい。

【００６０】

図９や図１０の実行タスク決定部１２がタスクの切替を行うと、切替後のタスクのタスクIDが実行タスク決定部１２からレジスタセット選択部２２に通知される。レジスタセット選択部２２は、図１１のテーブル等を用いて、切替後のタスクに対応するレジスタセットIDを取得し、そのIDに対応するレジスタセットの値を演算部２３に供給する。演算部２３は、レジスタセット選択部２２で選択され

たレジスタセットの値を各レジスタに設定して演算処理を行い、その演算処理結果をレジスタセット選択部２２で選択されたレジスタセットに格納する。

【００６１】

このように、第５の実施形態では、タスクの切替が起きると、レジスタセットも切り替えるようにしたため、タスク切替時の処理準備時間を短縮でき、タスクの高速切替を実現できる。また、いったんタスクの処理を中断した後に再開する場合も、再開前にレジスタセットの値を読み込めばよいため、いったん中断したタスクの処理を迅速に再開することができる。

【００６２】

（第６の実施形態）

第６の実施形態は、レジスタセット群２１の中の少なくとも一部のレジスタセットを外部に退避するものである。

【００６３】

図１２は本発明に係るプロセッサの第６の実施形態の概略構成を示すブロック図である。図１２のプロセッサは、図１０の構成に加えて、ヒット判定部３１と、外部レジスタセット記憶部３２と、外部記憶制御部３３と、レジスタセット転送部３４と、を有する。

【００６４】

本実施形態のレジスタセット群２１は、時分割で実行するタスクの数ｎより多くても少なくても等しくても構わない。

【００６５】

ヒット判定部３１は、実行すべきタスクのレジスタセットがレジスタセットIDタスクID対応部２４に登録されているか否かを判定する。レジスタセット転送部３４は、レジスタセット群２１の中の少なくとも一部のレジスタセットの内容を外部レジスタセッド記憶部３２に転送するとともに、外部レジスタセット記憶部３２から読み出したレジスタセットの内容をレジスタセット群２１に転送する。外部レジスタセット記憶部３２は、レジスタセット群２１の中の少なくとも一部のレジスタセットの内容を記憶する。

【００６６】

図１３は外部レジスタセット記憶部３２のデータ構成を示す図である。外部レジスタセット記憶部３２は、レジスタセット群２１に含まれる任意のレジスタセットの内容を記憶でき、また記憶したレジスタセットの内容をレジスタセット群２１に転送することもできる。外部レジスタセット記憶部３２に記憶された各レジスタセットはタスクIDで管理され、いったん記憶されたレジスタセットの内容を呼び出す際も、タスクIDを指定する。

【００６７】

このように、外部レジスタセット記憶部３２は、レジスタセット群２１の中の少なくとも一部のレジスタセットの内容を一時的に退避し、必要に応じてレジスタセット群２１に転送することができる。

【００６８】

外部レジスタセット記憶部３２は、専用のハードウェアで構成してもよいし、主記憶メモリなどの予め設けられているメモリの一部領域を利用してもよい。

【００６９】

図１４はヒット判定部３１の処理手順を示すフローチャートである。まず、実行すべきタスクを表すタスクIDに対応するレジスタセットIDがレジスタセットIDタスクID対応部２４に登録されているか否かを判定する（ステップＳ１１）。登録されていれば、ヒットしたと判定して、該当するタスクIDに対応するレジスタセットIDを取得する（ステップＳ１２）。この場合、図５と同様の手順でタスク切替を行う。

【００７０】

一方、登録されていなければ、ミスしたと判定し（ステップＳ１３）、外部レジスタセット記憶部３２から該当するレジスタセットを呼び出して、レジスタセット群２１の一部の入れ替え処理を行う。

【００７１】

より具体的には、まず、レジスタセット群２１の中の少なくとも一部のレジスタセットの内容をレジスタセット転送部３４を介して外部レジスタセット記憶部３２に転送する。同時に、実行すべきタスクが利用するレジスタセットの内容を外部レジスタセット記憶部３２から読み出して、レジスタセット転送部３４を介

してレジスタセット群２１に転送する。

【００７２】

このように、第６の実施形態では、レジスタセット群２１の中の少なくとも一部のレジスタセットの内容を外部レジスタセット記憶部３２に退避し、必要に応じて外部レジスタセット記憶部３２からレジスタセット群２１にレジスタセットの内容を戻せるようにしたため、レジスタセット群２１の中のレジスタセットの総数以上のタスクを処理することができる。したがって、レジスタセット群２１の中のレジスタセットの数を削減でき、プロセッサのサイズを小型化できる。

【００７３】

（第７の実施形態）

第７の実施形態は、ヒット判定部３１でヒットしなかったときに、退避させるべきレジスタセットを予め指定するものである。

【００７４】

図１５は本発明に係るプロセッサの第７の実施形態の概略構成を示すブロック図である。図１５のプロセッサは、図１２の構成に加えて、退避レジスタセット決定部３５を備えている。

【００７５】

退避レジスタセット決定部３５は、タスクID・優先順位供給部１４から供給された優先順位に基づいて、外部レジスタセット記憶部３２に退避させるべきレジスタセットをレジスタセット群２１の中から決定する。レジスタセット転送部３４は、退避レジスタセット決定部３５で決定されたレジスタセットの内容を外部レジスタセット記憶部３２に転送するとともに、外部レジスタセット記憶部３２から読み出したレジスタセットの内容をレジスタセット群２１に転送する。

【００７６】

図１６は退避レジスタセット決定部３５の処理手順を示すフローチャートである。まず、レジスタセット群２１の中に利用するレジスタセットが存在し、かつそのレジスタセットに対応するタスクが実行不可能であるか否かを判定する（ステップＳ２１）。判定がYesの場合、該当するタスクの中で最も優先順位の低いタスクを選択する（ステップＳ２２）。

【００７７】

一方、ステップＳ２１の判定がNoの場合、レジスタセット群２１の中の各レジスタセットに対応するタスクのうち、最も優先順位の低いタスクを選択する（ステップＳ２３）。

【００７８】

ステップＳ２２またはＳ２３の処理が終了すると、選択されたタスクが利用するレジスタセットのID（レジスタセットID）を取得し（ステップＳ２４）、取得したIDをレジスタセット転送部３４に指示する（ステップＳ２５）。

【００７９】

このように、第７の実施形態では、外部レジスタセット記憶部３２に退避すべきレジスタセットを明示的に指定できるようにしたため、使用頻度の低いレジスタセットを入れ替えることができ、レジスタセットの退避による処理効率の低下を防止できる。

【００８０】

（第８の実施形態）

上述した第６及び第７の実施形態は、実行すべきタスクが利用するレジスタセットがレジスタセット群２１に存在するか否かをヒット判定部３１で判定する処理を行うため、タスクの切替を迅速に行えないおそれがある。そこで、以下に説明する第８の実施形態は、ヒット判定部３１が処理を行っている間に、ヒット判定部３１の判定結果の影響を受けないタスクの処理を行うものである。

【００８１】

図１７は本発明に係るプロセッサの第８の実施形態の概略構成を示すブロック図である。図１７の実行タスク決定部１２は、図１５の実行タスク決定部１２とは異なる処理を行う。すなわち、図１７の実行タスク決定部１２は、ヒット判定部３１に対して判定結果の送信を要求し、ヒット判定部３１からの判定結果を受信する。

【００８２】

図１７のレジスタセットIDタスクID対応部２４は、図１８に示すように、個々のタスクID及びレジスタセットIDに対応づけて、フラグを有する。このフラグは

、あるタスクを実行しようとしたが、ヒット判定部３１でミスし、そのタスクが利用するレジスタセットの内容を外部レジスタセット記憶部３２から転送する処理を行っている最中にセットされる。

【００８３】

実行タスク決定部１２は、実行するタスクを切り替える際に、ミス時転送ヒット判定要求またはヒット判定要求のいずれかを出す。ミス時転送ヒット判定要求は、ヒット判定部３１に対してタスクIDを通知し、ミスした場合には、そのタスクが利用するレジスタセットを外部レジスタセット転送装置からレジスタセット群２１に転送するようレジスタセット選択部２２に指示するものである。

【００８４】

ミス時転送ヒット判定要求を受けた場合のヒット判定部３１の処理手順は図１９のフローチャートで表される。まず、実行すべきタスクのタスクIDがレジスタセットIDタスクID対応部２４に登録されているか否かを判定する（ステップＳ３１）。登録されていれば、該当するタスクIDに対応するレジスタセットIDを取得し（ステップＳ３２）、登録されていなければ、ミスと判定して（ステップＳ３３）、該当するタスクが利用するレジスタセットの転送をレジスタセット転送部３４に指示する（ステップＳ３４）。

【００８５】

一方、ヒット判定要求は、ヒット判定部３１に対してタスクIDを通知し、フラグが有効であるレジスタセットを除外してヒット判定を行うものである。ミスした場合でもレジスタセットの転送は行わない。

【００８６】

ヒット判定要求を受けた場合のヒット判定部３１の処理手順は図２０のフローチャートで表される。まず、フラグがセットされているタスクIDを除いて、実行すべきタスクのタスクIDがレジスタセットIDタスクID対応部２４に登録されているか否かを判定する（ステップＳ４１）。登録されていれば、そのタスクIDに対応するレジスタセットIDを取得し（ステップＳ４２）、登録されていなければ、ミスしたと判定する（ステップＳ４３）。

【００８７】

レジスタセットIDタスクID対応部２４の中に、セットされているフラグが存在しない場合には、実行タスク決定部１２は、実行するタスクを切り替える際に、切替後のタスクIDでミス時転送ヒット判定要求を出す。ヒットした場合には、実行タスク決定部１２は、ヒットした新しいタスクIDに対応するレジスタセットIDをレジスタセットIDタスクID対応部２４から取得し、そのレジスタセットIDをレジスタセット選択部２２に通知し、新しいタスクの実行を開始する。一方、ミスした場合は、ヒット判定部３１はレジスタセット転送部３４にレジスタセットの転送を指示し、レジスタセットIDタスクID対応部２４は、転送を行うレジスタセットに対応するフラグをセットする。

【００８８】

レジスタセットの転送が終了すると、レジスタセット転送部３４は、レジスタセットIDタスクID対応部２４に転送完了を通知する。この通知に従って、レジスタセットIDタスクID対応部２４はレジスタセットのフラグを無効化する。

【００８９】

実行タスク決定部１２は、実行すべきタスクに対応するレジスタセットIDをレジスタセットIDタスクID対応部２４から取得し、取得したレジスタセットIDをレジスタセット選択部２２に通知して、そのタスクの実行を開始する。

【００９０】

レジスタセットIDタスクID対応部２４の中に、セットされているフラグが存在する場合には、実行タスク決定部１２は、実行するタスクを切り替える際に、切替後のタスクIDでヒット判定要求を出す。ヒットした場合には、実行タスク決定部１２は、ヒットした新しいタスクIDに対応するレジスタセットIDをレジスタセットIDタスクID対応部２４から取得し、そのレジスタセットIDをレジスタセット選択部２２に通知し、新しいタスクの実行を開始する。一方、ミスした場合には、タスクの切替を行わない。

【００９１】

図２１はレジスタセットIDタスクID対応部２４の中にセットされているフラグが存在する場合のタスクの実行形態を説明する図である。この例では、タスクID5のタスクを実行中に、このタスクよりも優先順位の高いタスクID1のタスクの実

行が可能になったときに、実行タスク決定部１２はヒット判定部３１に対してタスクID1のミス時転送ヒット判定要求を行う。

【００９２】

この例では、この要求を受けたヒット判定部３１はミス判定を行っており、これにより、レジスタセット転送部３４はタスクID1が利用するレジスタをレジスタセット群２１で転送を開始する。この転送には、時刻ｔ３までの時間がかかる。

【００９３】

時刻ｔ１～ｔ３の間の時刻ｔ２でタスクID5の実行が終了すると、実行タスク決定分は実行可能なタスクの中で最も優先順位の高いタスク（図２１の例ではタスクID7）のヒット判定要求を行う。ヒットした場合には、タスクID7の実行を行う。

【００９４】

その後、時刻ｔ３になると、レジスタセット転送部３４からの転送完了が通知され、それによって実行タスク決定分はタスクID1の実行を開始する。

【００９５】

このように、第８の実施形態では、あるタスクが利用するレジスタセットの転送中に、そのタスクの実行に影響しない他のタスクの実行を行うようにしたため、レジスタセットの転送完了を待つことなくダスクの処理を行え、タスクの処理効率を向上できる。

【００９６】

（第９の実施形態）

第９の実施形態は、レジスタセットを構成する各レジスタに、レジスタの内容が更新されたか否かを示すフラグを設けるものである。

【００９７】

第９の実施形態は、図１２と同様のブロック構成を有するが、レジスタセット群２１のレジスタセットのデータ構造が図１２とは異なっている。図２２は第９の実施形態のレジスタセットのデータ構造を示す図である。図示のように、レジスタセットを構成する各レジスタに対応して変更フラグが設けられている。この

変更フラグは、対応するレジスタの内容を書き換えるときにセットされる。この場合の処理手順は図２３のようなフローチャートで表される。

【００９８】

図２３において、まず、書き換えるレジスタに対応する変更フラグをセットし（ステップＳ５１）、その後にレジスタを書き換える（ステップＳ５２）。

【００９９】

図２２の変更フラグは、外部レジスタセット記憶部３２に記憶されているレジスタセットを読み込むときにクリアされる。外部レジスタセット記憶部３２からレジスタセット群２１への転送要求があった場合の処理手順は図２４のようなフローチャートで表される。

【０１００】

図２４において、まず、転送先のレジスタセット内の変更フラグをすべてクリアし（ステップＳ６１）、その後にレジスタセットを転送する（ステップＳ６２）。

【０１０１】

図２５はレジスタセット群２１が外部レジスタセット記憶部３２にレジスタセットの内容の転送要求を行った場合のレジスタセット転送部３４の処理手順を示すフローチャートである。まず、転送するレジスタセット内の１つのレジスタについて注目し（ステップＳ７１）、注目したレジスタセットの変更フラグがセットされているか否かを判定する（ステップＳ７２）。

【０１０２】

セットされている場合には、転送するレジスタを外部レジスタセット記憶部３２に転送する（ステップＳ７３）。セットされていない場合か、ステップＳ７３の処理が終了した場合は、すべてのレジスタについて処理を行ったか否かを判定し（ステップＳ７４）、処理を行っていないものがあればステップＳ７１に戻り、処理を行った場合は終了する。

【０１０３】

このように、第９の実施形態では、レジスタセット群２１の中に含まれるレジスタのうち、変更のあったものだけを外部レジスタセット記憶部３２に転送する

ようにしたため、転送すべきデータ量を削減でき、転送時間の削減が図れる。

【０１０４】

（第１０の実施形態）

第１０の実施形態は、外部レジスタセット記憶部３２の中に、各レジスタセットの各レジスタが書き換えられたか否かを示すフラグを有する。

【０１０５】

図２６は外部レジスタセット記憶部３２のデータ構造を示す図である。図示のように、各レジスタセットの各レジスタに対応して、有効フラグが設けられている。この有効フラグは、対応するレジスタが書き換えられた場合にセットされる。

【０１０６】

図２７は有効フラグの初期化処理を示すフローチャートであり、タスクの起動時に外部記憶制御部３３が行う処理である。外部記憶制御部３３は、起動するタスクに対応するレジスタセット内の有効フラグすべてをクリアする（ステップＳ８１）。

【０１０７】

図２８は有効フラグのセット処理を示すフローチャートであり、レジスタセット転送部３４から外部レジスタセット記憶部３２へのレジスタセットの転送要求があったときに、外部記憶制御部３３が行う処理である。外部記憶制御部３３は、外部レジスタセット記憶部３２の転送するレジスタに対応する有効フラグをセットする（ステップＳ９１）。その後、レジスタセット群２１から外部レジスタセット記憶部３２への該当レジスタの転送を行う（ステップＳ９２）。

【０１０８】

図２９は外部レジスタセット記憶部３２からレジスタセット群２１への転送要求があった場合の外部記憶制御部３３の処理手順を示すフローチャートである。まず、転送するレジスタセット内の有効フラグを読み込む（ステップＳ１０１）。次に、転送するレジスタセット内の１つのレジスタについて着目する（ステップＳ１０２）。

【０１０９】

このレジスタに対応する有効フラグがセットされているか否かを判定し（ステップＳ１０３）、セットされていれば、レジスタを外部レジスタセット記憶部３２に転送する（ステップＳ１０４）。次に、すべてのレジスタについて処理を行ったか否かを判定し（ステップＳ１０５）、まだ処理を行っていないレジスタがあれば、ステップＳ１０１以降の処理を繰り返す。

【０１１０】

このように、第１０の実施形態では、外部レジスタセット記憶部３２の中に、レジスタの書き換えを行ったか否かを示す有効フラグを設けるため、レジスタの内容が書き換えられた場合のみ、外部レジスタセット記憶部３２からレジスタセット群２１にレジスタの内容を転送すればよく、転送回数を削減できるとともに、転送時間も短縮できる。

【０１１１】

（第１１の実施形態）

第１１の実施形態は、外部レジスタセット記憶部３２の各タスクのレジスタセットをリスト構造にするものである。

【０１１２】

図３０は第１１の実施形態の外部レジスタセット記憶部３２のデータ構造を示す図である。外部レジスタセット記憶部３２は、リスト構造になっており、変更のあったレジスタの値を記憶するレジスタ値記憶領域と、レジスタ値記憶領域に記憶されているレジスタの識別番号（レジスタID）とを組にして記憶する。すなわち、外部レジスタセット記憶部３２は、内容に変更のないレジスタの値は記憶しない。

【０１１３】

図３１は外部レジスタセット記憶部３２からレジスタセット群２１への転送要求があった場合の外部記憶制御部３３の処理手順を示すフローチャートである。まず、転送するレジスタセットのリストの先頭にポインタを移動する（ステップＳ１１１）。次に、外部レジスタセット記憶部３２のリストからレジスタIDとレジスタの値を読み込む（ステップＳ１１２）。次に、レジスタIDに対応するレジスタセット群２１のレジスタにレジスタの値を書き込む（ステップＳ１１３）。

次に、リストの最後か否かを判定する（ステップＳ１１４）。リストの最後であれば処理を終了し、リストの最後でなければ、リストの次の項目に移動してステップＳ１１１以降の処理を繰り返す。

【０１１４】

図３２はレジスタセット転送部３４から外部レジスタセット記憶部３２にレジスタの転送要求があった場合の外部記憶制御部３３の処理手順を示すフローチャートである。まず、転送するレジスタIDが外部レジスタセット記憶部３２の対応するリスト中にあるか否かを判定する（ステップＳ１２１）。リスト中になければ、リストに転送するレジスタIDの項目を追加する（ステップＳ１２２）。

【０１１５】

ステップＳ１２１でリスト中にあると判定された場合か、ステップＳ１２２の処理が終了した場合は、リストの転送するレジスタIDに対応するレジスタ値記憶領域に、レジスタの値を書き込む（ステップＳ１２３）。

【０１１６】

図３３はタスク終了時の外部記憶制御部３３の処理手順を示すフローチャートである。外部レジスタセット記憶部３２の中の終了したタスクに対応するリストの内容を破棄する（ステップＳ１３１）。

【０１１７】

このように、第１１の実施形態では、外部レジスタセット記憶部３２をリスト構造にし、変更のあったレジスタの値のみを記憶するようにしたため、外部レジスタセット記憶部３２のメモリ容量を削減できるとともに、レジスタセット群２１との間で転送されるデータ量も削減できる。

【０１１８】

上述した第１～第１１の実施形態では、本発明をプロセッサに適用した例を説明したが、本発明はプロセッサ以外の半導体集積装置にも適用可能である。例えば、図１等の実行タスク決定部１２のみをプロセッサとは別個に設けてもよい。

【０１１９】

【発明の効果】

以上詳細に説明したように、本発明によれば、第１及び第２のFIFO型記憶手段

のデータ格納状態によりタスクの起動準備が整ったか否かを判断し、起動準備が整ったタスクのみ起動するようにしたため、オペレーティングシステムなしでタスクの起動スケジューリングを行うことができ、オペレーティングシステムのオーバーヘッドがなくなって、複数のタスクを効率よく処理することができる。

【０１２０】

また、タスクで処理するデータを格納する第１及び第２のFIFO型記憶手段を監視するため、データを消失するおそれがなくなり、信頼性の高いタスク処理を行うことができる。

【図面の簡単な説明】

【図１】

ソフトウェア無線機の処理内容を示す図。

【図２】

タスク間にFIFOを設けたプロセッサの概略構成図。

【図３】

本発明に係るプロセッサの第１の実施形態の概略構成を示すブロック図。

【図４】

本発明に係るプロセッサの第２の実施形態の概略構成を示すブロック図。

【図５】

実行タスク決定部１２とタスクID・優先順位供給部１４の詳細構成を示すブロック図。

【図６】

図５の実行タスク決定部１２の処理手順を示すフローチャート。

【図７】

本発明に係るプロセッサの第３の実施形態の概略構成を示すブロック図。

【図８】

本発明に係るプロセッサの第４の実施形態の動作を説明するタイムチャート。

【図９】

本発明に係るプロセッサの第５の実施形態の概略構成を示すブロック図。

【図１０】

図９の変形例を示すプロセッサのブロック図。

【図１１】

レジスタセットIDタスクID対応部２４のデータ構造を示す図。

【図１２】

本発明に係るプロセッサの第６の実施形態の概略構成を示すブロック図。

【図１３】

外部レジスタセット記憶部３２のデータ構成を示す図。

【図１４】

ヒット判定部３１の処理手順を示すフローチャート。

【図１５】

本発明に係るプロセッサの第７の実施形態の概略構成を示すブロック図。

【図１６】

退避レジスタセット決定部３５の処理手順を示すフローチャート。

【図１７】

本発明に係るプロセッサの第８の実施形態の概略構成を示すブロック図。

【図１８】

第８の実施形態のレジスタセットIDタスクID対応部のデータ構造を示す図。

【図１９】

ミス時転送ヒット判定要求を受けた場合のヒット判定部の処理手順を示すフローチャート。

【図２０】

ヒット判定要求を受けた場合のヒット判定部の処理手順を示すフローチャート。

【図２１】

レジスタセットIDタスクID対応部２４の中にセットされているフラグが存在する場合のタスクの実行形態を説明する図。

【図２２】

第９の実施形態のレジスタセットのデータ構造を示す図。

【図２３】

レジスタの書き換え要求があった場合の処理手順を示すフローチャート。

【図２４】

外部レジスタセット記憶部からレジスタセット群への転送要求があった場合の処理手順を示すフローチャート。

【図２５】

レジスタセット群から外部レジスタセット記憶部への転送要求があった場合の処理手順を示すフローチャート。

【図２６】

外部レジスタセット記憶部３２のデータ構造を示す図。

【図２７】

タスク起動時の処理手順を示すフローチャート。

【図２８】

レジスタセット転送部から外部レジスタセット記憶部へのレジスタ転送要求があった場合の処理手順を示すフローチャート。

【図２９】

外部レジスタセット記憶部からレジスタセット群への転送要求があった場合の処理手順を示すフローチャート。

【図３０】

第１１の実施形態の外部レジスタセット記憶部３２のデータ構造を示す図。

【図３１】

外部レジスタセット記憶部からレジスタセット群への転送要求があった場合の処理手順を示すフローチャート。

【図３２】

レジスタセット転送部から外部レジスタセット記憶部へのレジスタ転送要求があった場合の処理手順を示すフローチャート。

【図３３】

タスク終了処理を示すフローチャート。

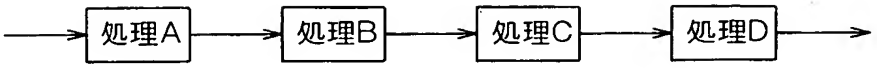
【符号の説明】

１１　タスク起動部

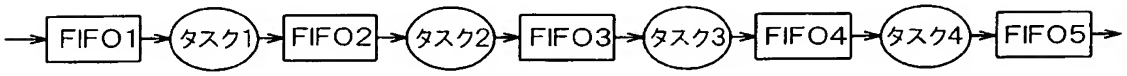
１２　実行タスク決定部
１３　プロセッサコア
１４　タスクID・優先順位供給部
１５　実行可能タスクIDリスト
１６　プライオリティエンコーダ
１７　時刻計測部
１８　起動時刻テーブル
１９　平均時間感覚計算部
２０　優先順位決定部
２１　レジスタセット群
２２　レジスタセット選択部
２３　演算部
２４　レジスタセットIDタスクID対応部
３０　優先順位変更部
３１　ヒット判定部
３２　外部レジスタセット記憶部
３３　外部記憶制御部
３４　レジスタセット転送部
３５　退避レジスタセット決定部

【書類名】　図面

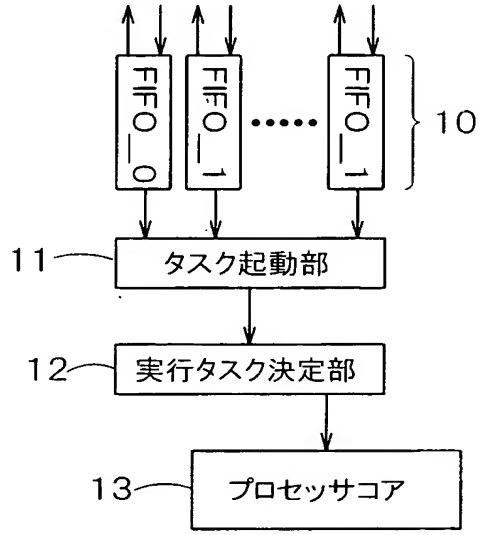
【図１】



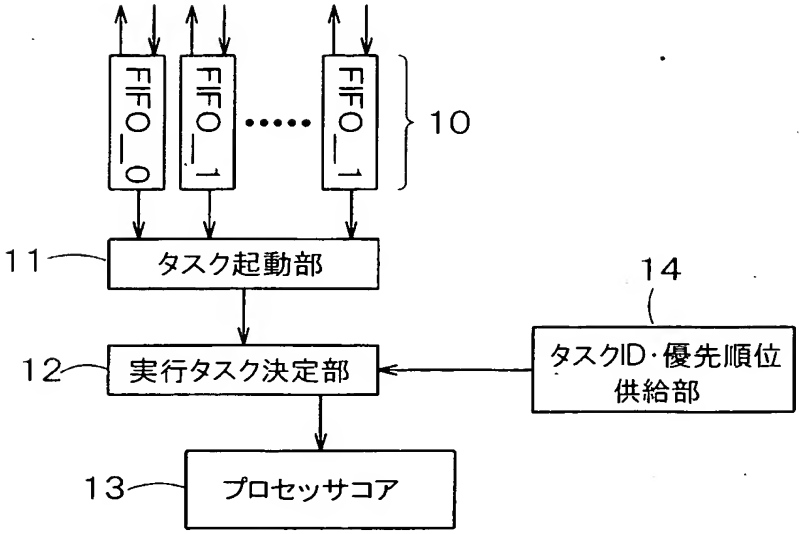
【図２】



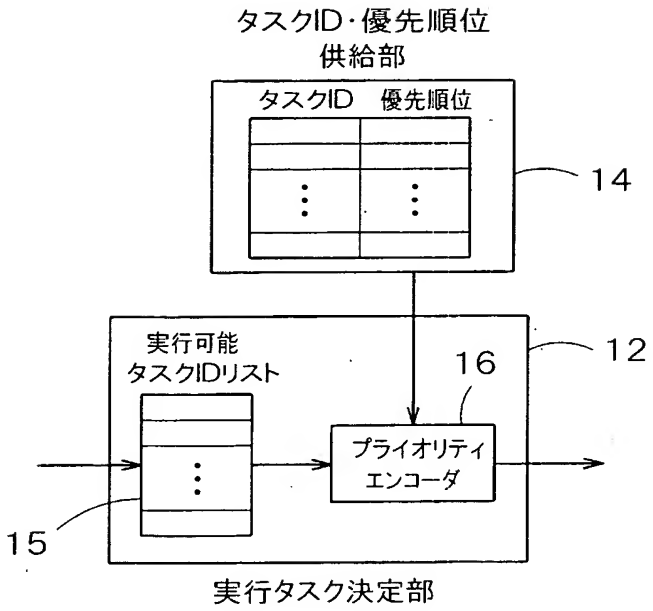
【図３】



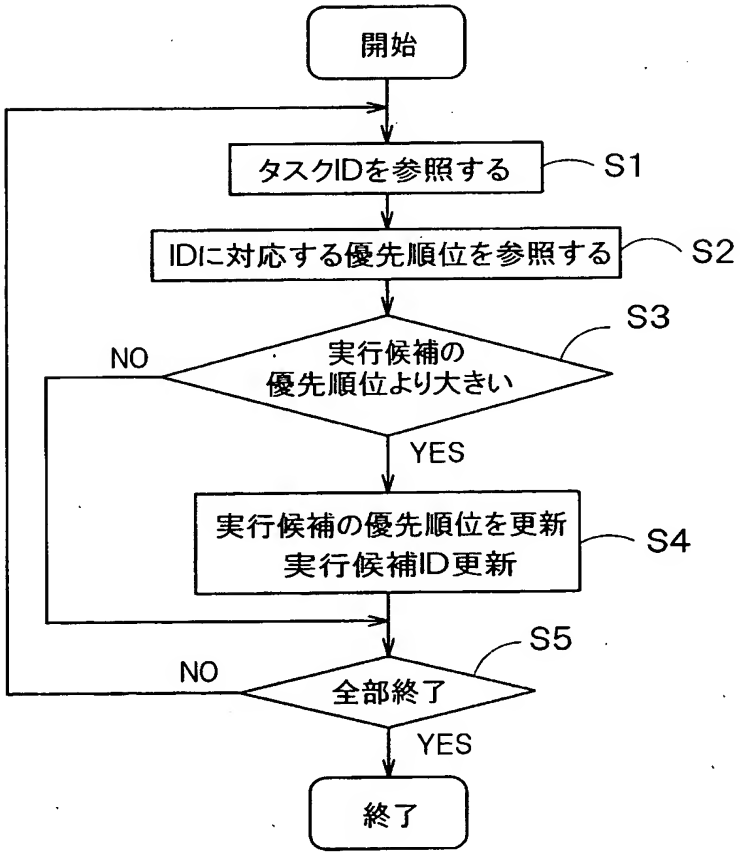
【図４】



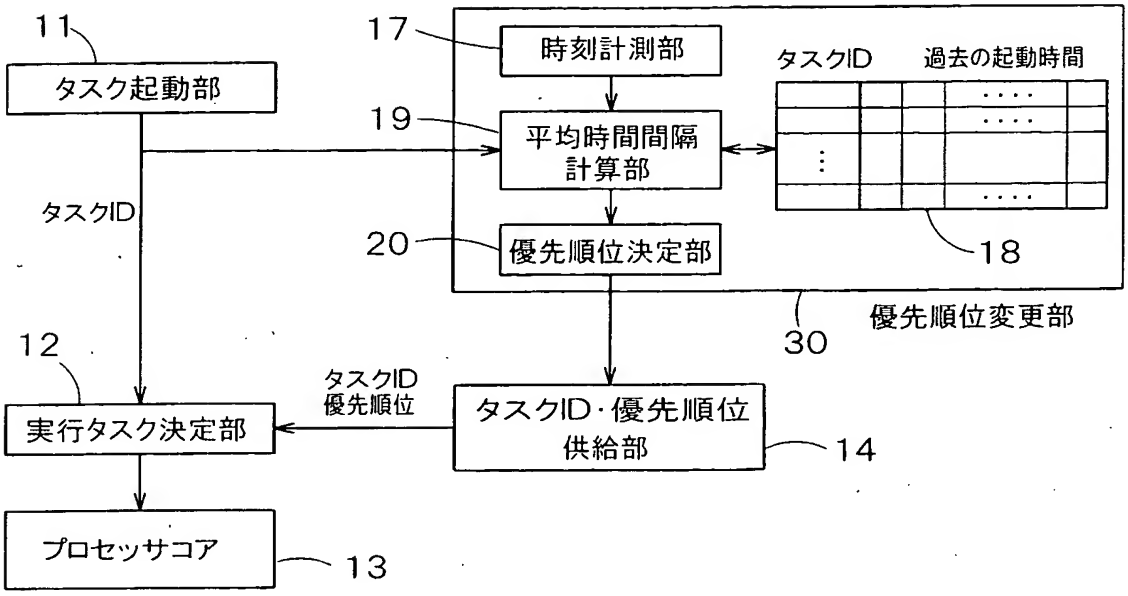
【図５】


タスクID・優先順位
供給部
タスクID　優先順位
14
実行可能
タスクIDリスト
16
12
プライオリティ
エンコーダ
15
実行タスク決定部

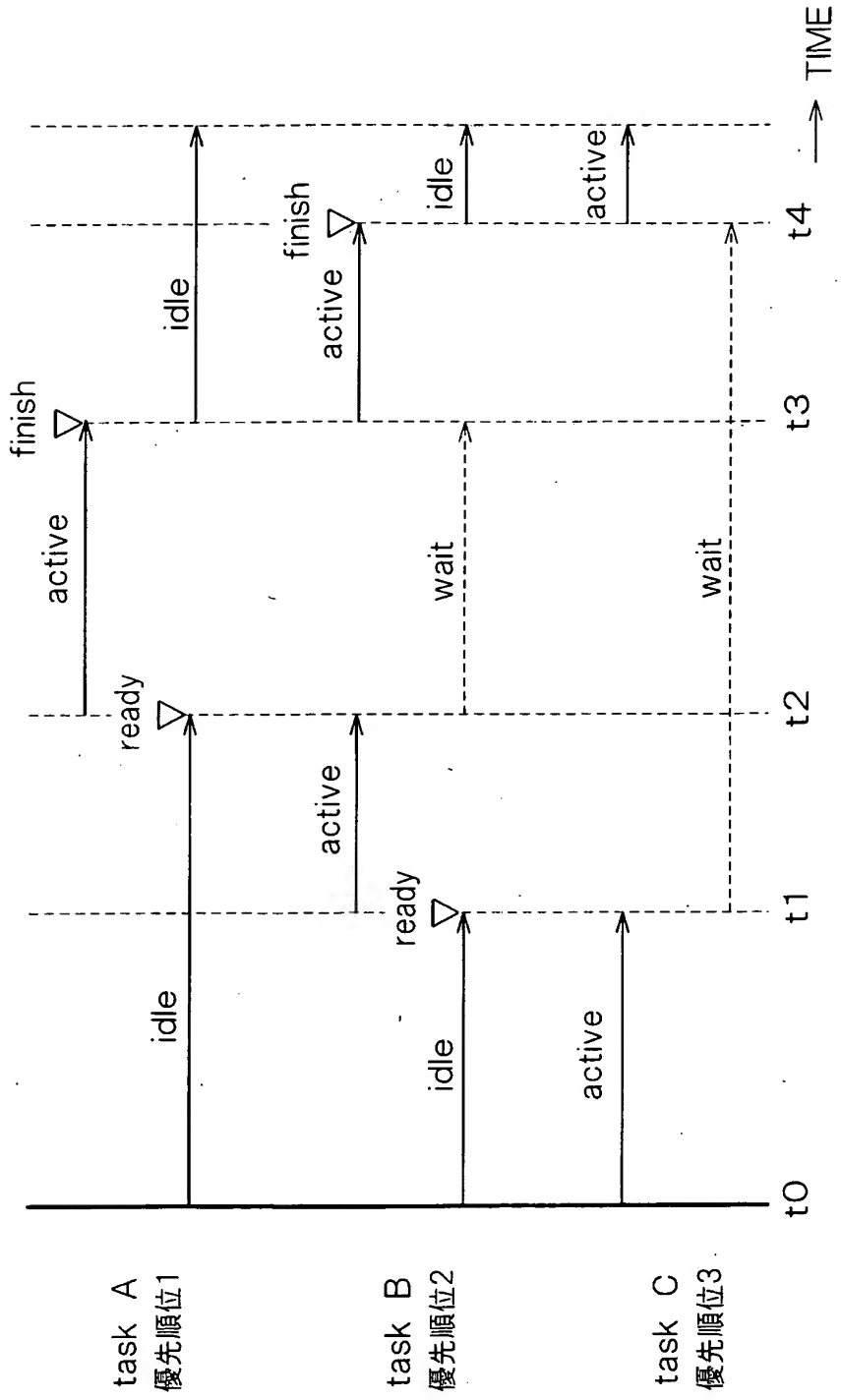
【図６】


開始
タスクIDを参照する
S1
IDに対応する優先順位を参照する
S2
S3
NO
実行候補の
優先順位より大きい
YES
実行候補の優先順位を更新
実行候補ID更新
S4
S5
NO
全部終了
YES
終了

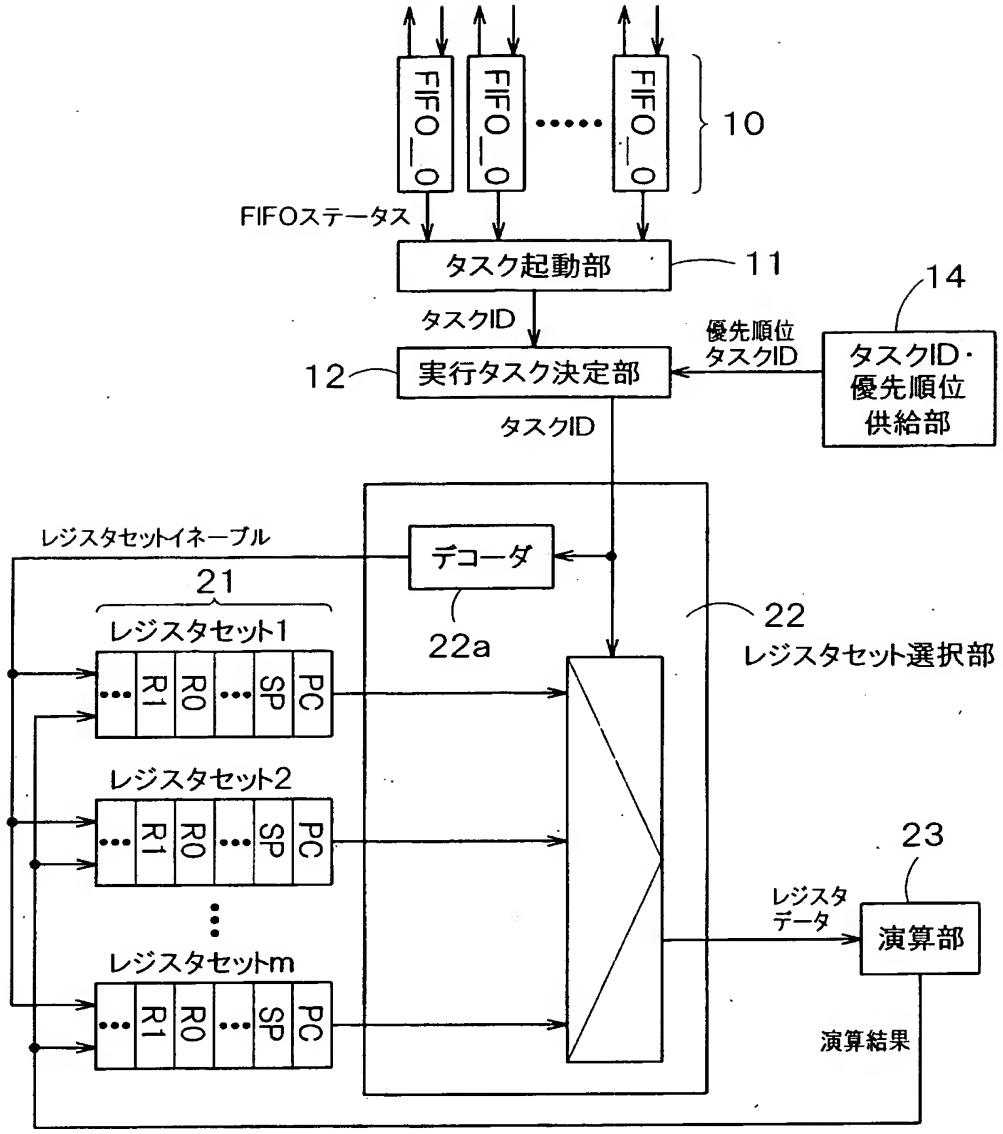
【図７】


11
タスク起動部
17
時刻計測部
19
平均時間間隔
計算部
20
優先順位決定部
タスクID
過去の起動時間
18
30
優先順位変更部
タスクID
12
実行タスク決定部
タスクID
優先順位
タスクID・優先順位
供給部
14
プロセッサコア
13

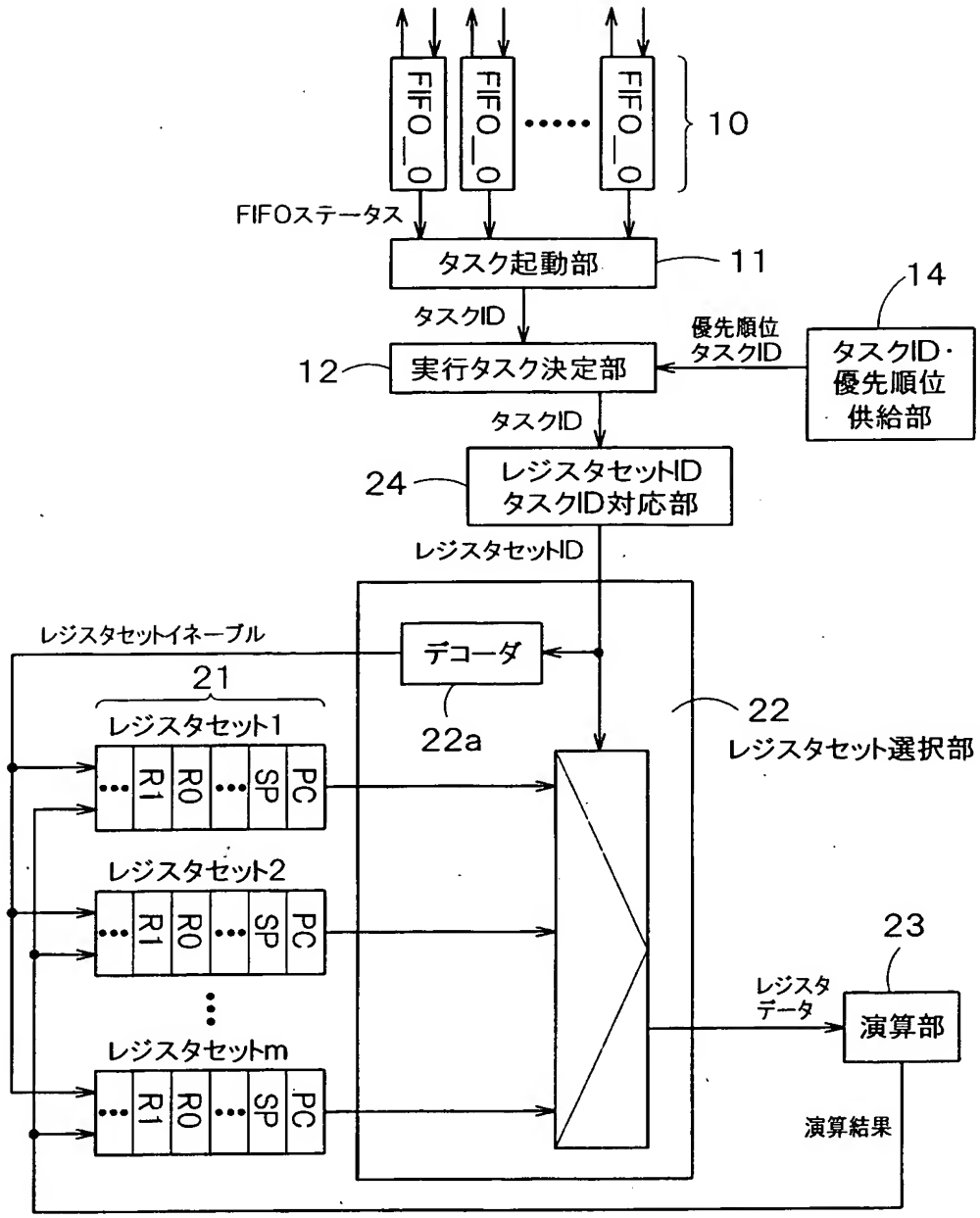
【図８】


TIME
t4
t3
t2
t1
t0
finish
active
idle
ready
wait
task A
優先順位1
task B
優先順位2
task C
優先順位3

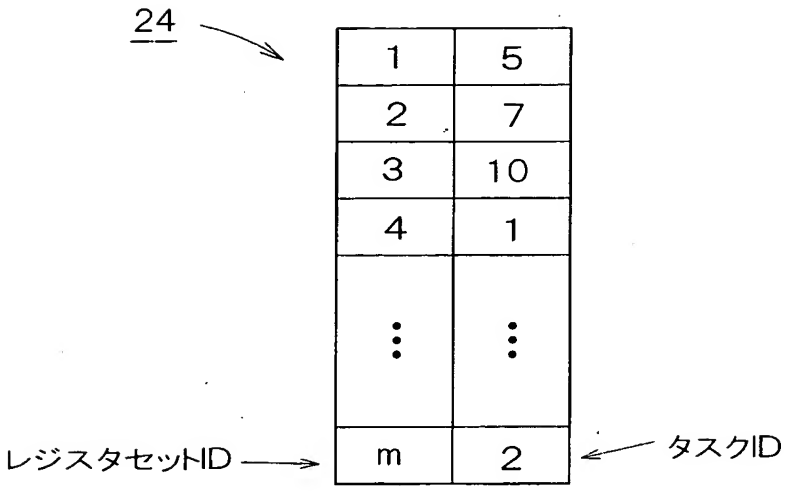
【図９】


FIFO_0
FIFO_0
FIFO_0
10
FIFOステータス
タスク起動部
11
14
タスクID
優先順位
タスクID
12
実行タスク決定部
タスクID・
優先順位
供給部
タスクID
レジスタセットイネーブル
デコーダ
21
22a
22
レジスタセット選択部
レジスタセット1
R1
R0
SP
PC
レジスタセット2
23
レジスタ
データ
演算部
レジスタセットm
演算結果

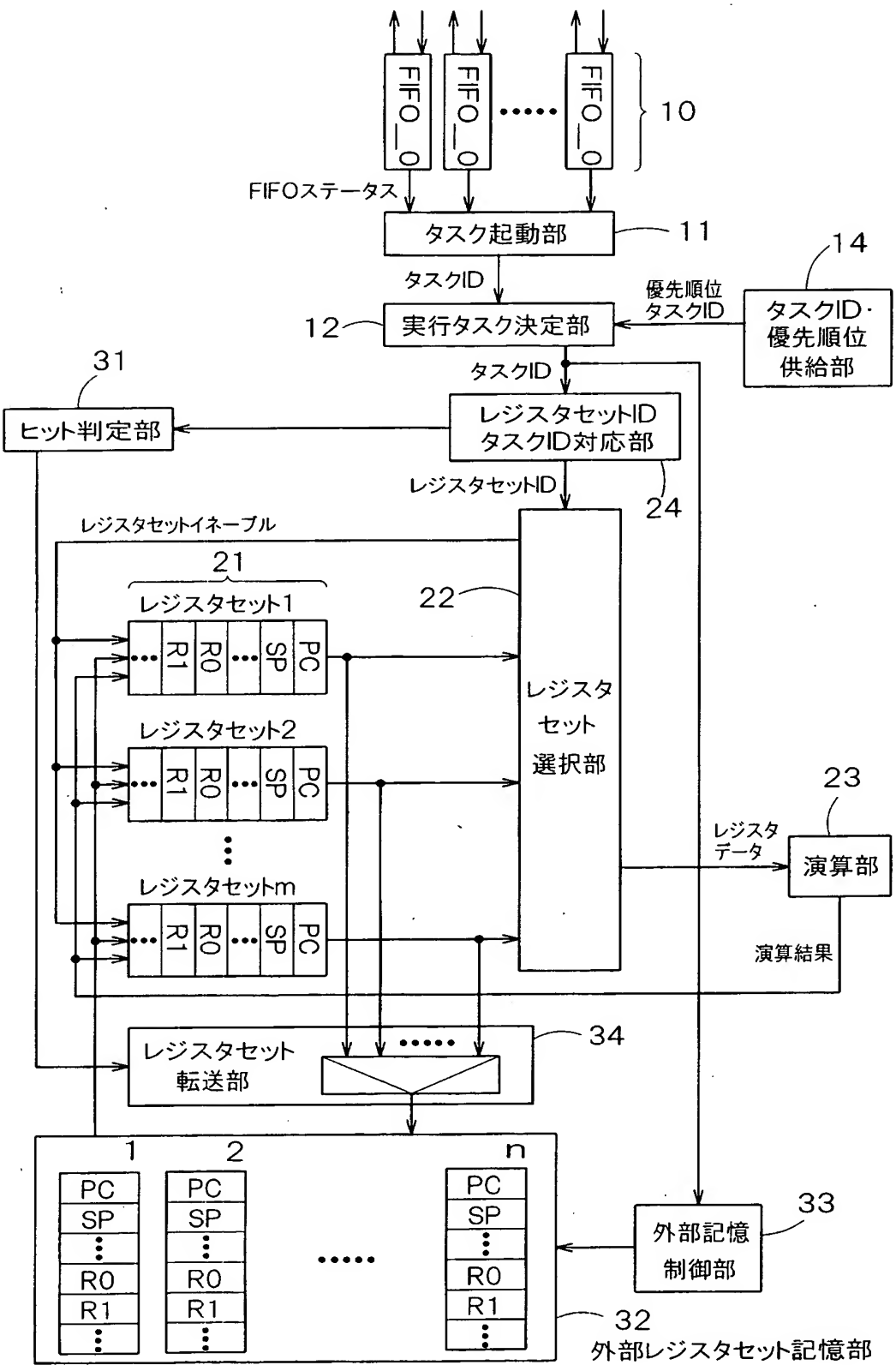
【図１０】


FIFO_0
FIFO_0
FIFO_0
10
FIFOステータス
タスク起動部
11
14
タスクID
優先順位
タスクID
12
実行タスク決定部
タスクID・
優先順位
供給部
タスクID
24
レジスタセットID
タスクID対応部
レジスタセットID
レジスタセットイネーブル
デコーダ
21
22a
22
レジスタセット選択部
レジスタセット1
R1
R0
SP
PC
レジスタセット2
23
レジスタ
データ
演算部
レジスタセットm
演算結果

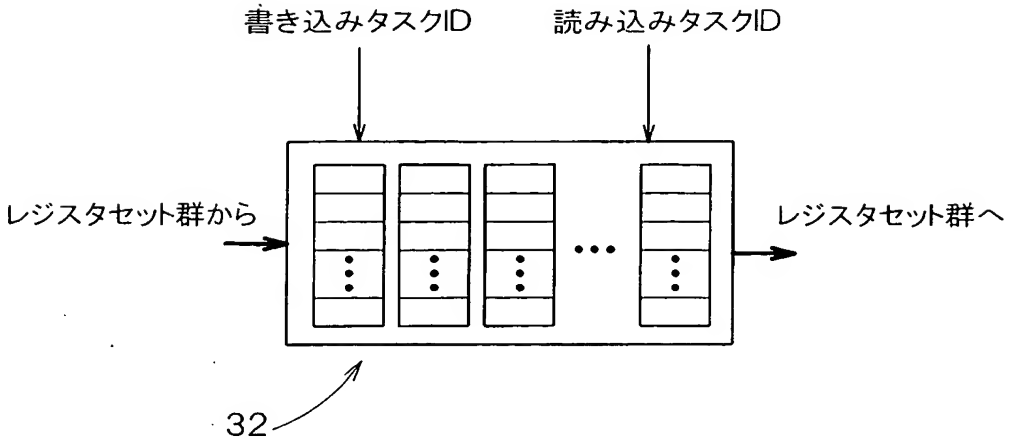
【図１１】



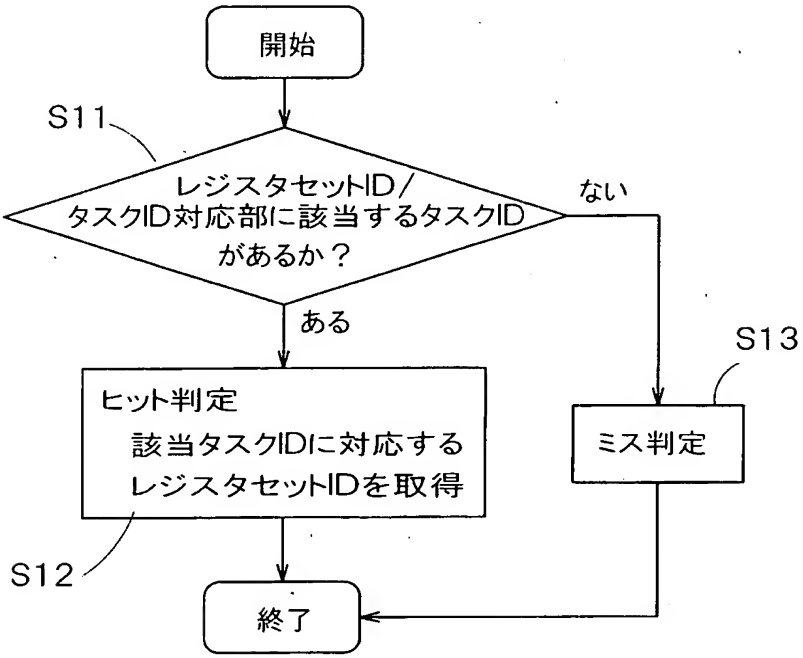
【図１２】


FIFO_0
FIFO_0
FIFO_0
10
FIFOステータス
タスク起動部
11
14
タスクID
優先順位
タスクID
12
実行タスク決定部
タスクID・
優先順位
供給部
タスクID
31
ヒット判定部
レジスタセットID
タスクID対応部
レジスタセットID
24
レジスタセットイネーブル
21
22
レジスタセット1
R1
R0
SP
PC
レジスタセット2
レジスタセットm
レジスタ
セット
選択部
23
レジスタ
データ
演算部
演算結果
レジスタセット
転送部
34
1
2
n
PC
SP
R0
R1
外部記憶
制御部
33
32
外部レジスタセット記憶部

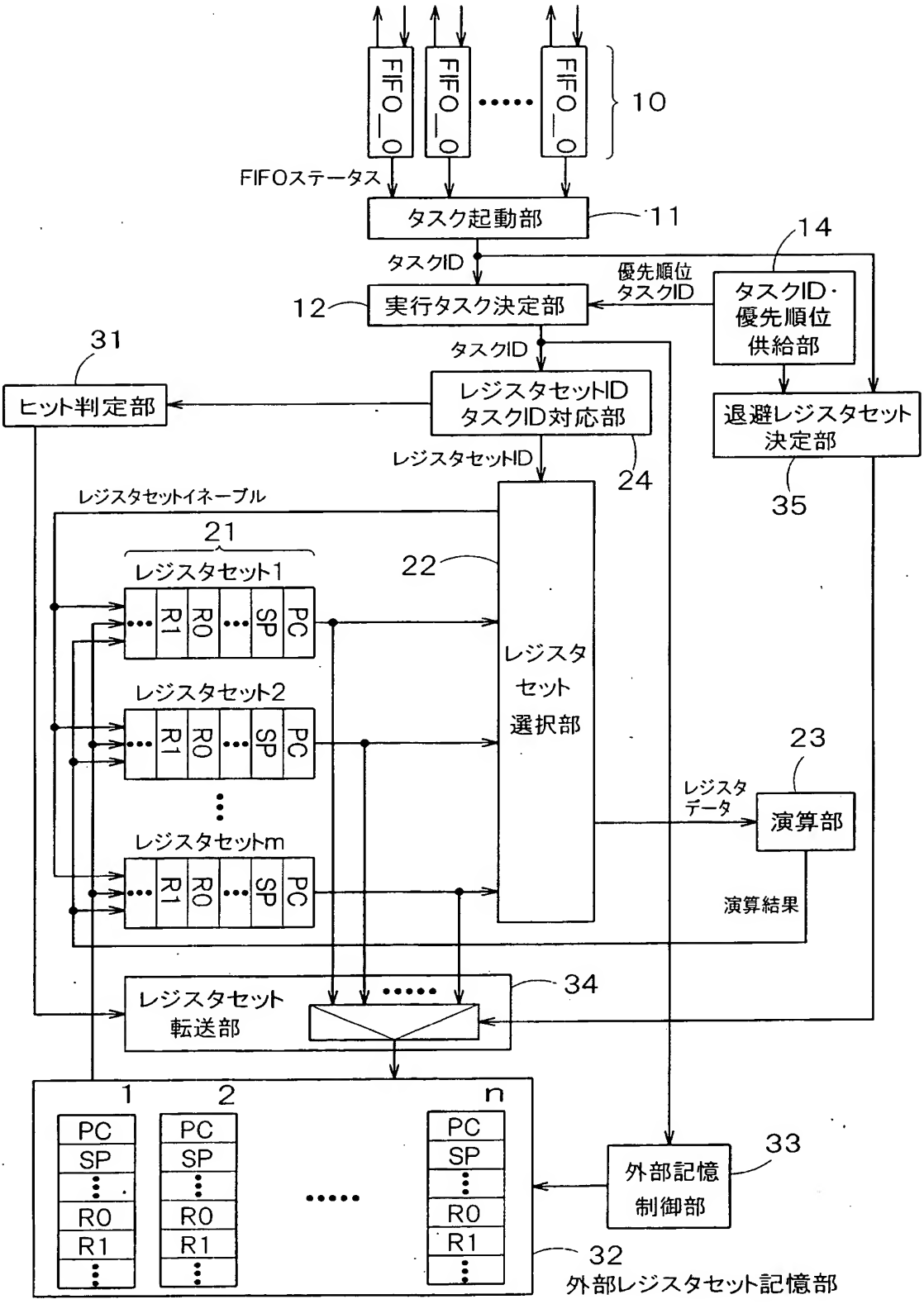
【図１３】


書き込みタスクID
読み込みタスクID
レジスタセット群から
レジスタセット群へ
32

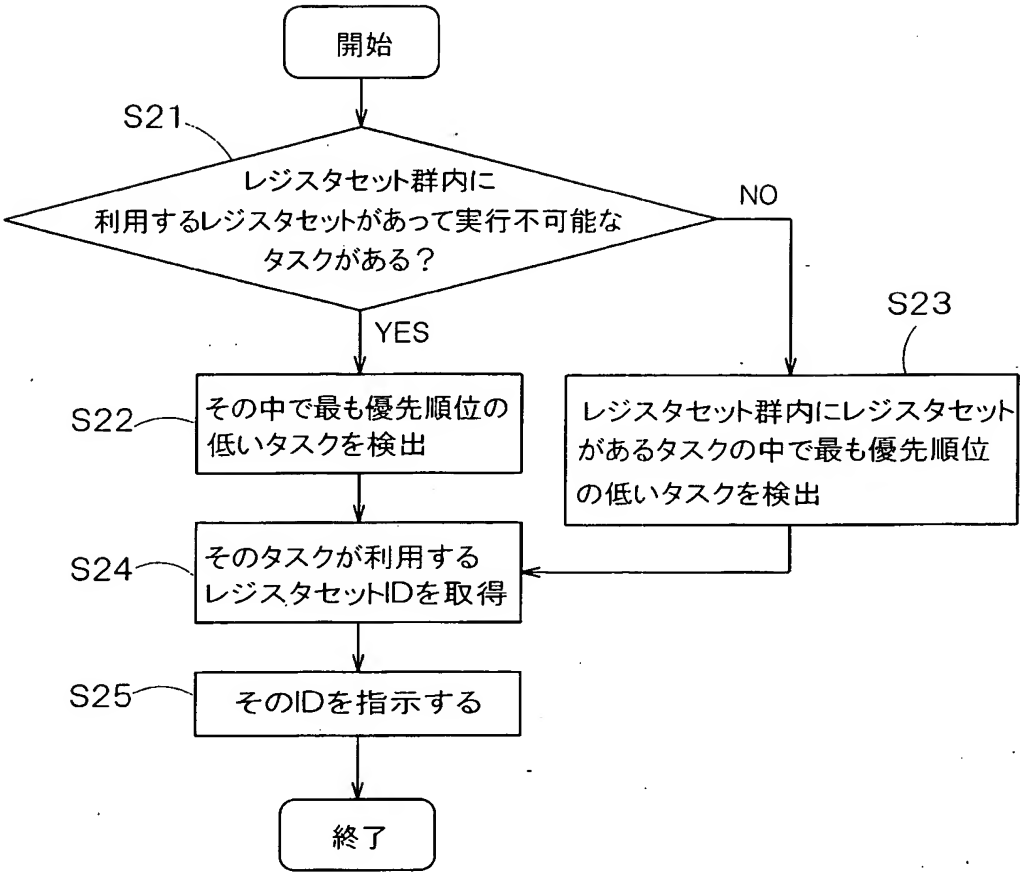
【図１４】


開始
S11
レジスタセットID／
タスクID対応部に該当するタスクID
があるか？
ない
ある
S13
ヒット判定
該当タスクIDに対応する
レジスタセットIDを取得
ミス判定
S12
終了

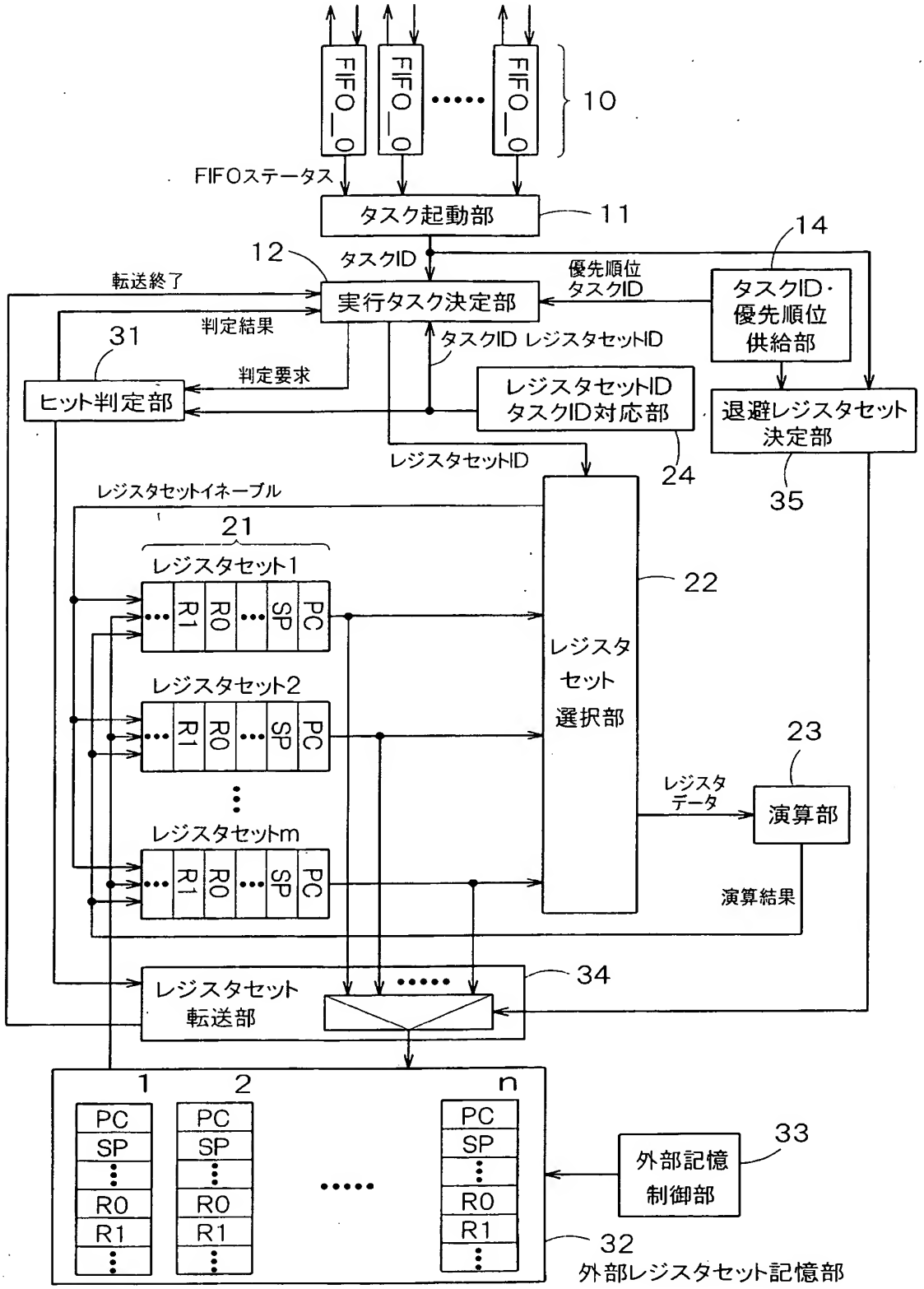
【図１５】


FIFO_0
FIFO_0
FIFO_0
10
FIFOステータス
タスク起動部
11
14
タスクID
優先順位
タスクID
12
実行タスク決定部
タスクID・
優先順位
供給部
31
タスクID
ヒット判定部
レジスタセットID
タスクID対応部
退避レジスタセット
決定部
レジスタセットID
24
35
レジスタセットイネーブル
21
22
レジスタセット1
R1
R0
SP
PC
レジスタセット2
レジスタ
セット
選択部
23
レジスタ
データ
演算部
レジスタセットm
演算結果
34
レジスタセット
転送部
1
2
n
PC
SP
R0
R1
33
外部記憶
制御部
32
外部レジスタセット記憶部

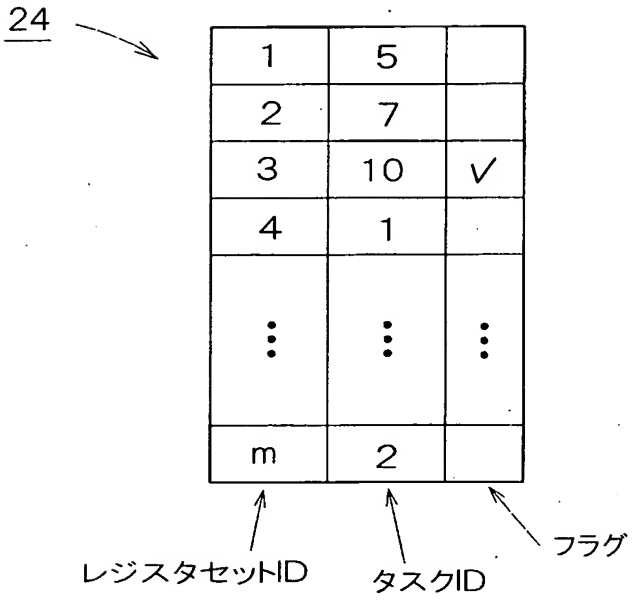
【図１６】


開始
S21
レジスタセット群内に
利用するレジスタセットがあって実行不可能な
タスクがある？
NO
YES
S23
S22
その中で最も優先順位の
低いタスクを検出
レジスタセット群内にレジスタセット
があるタスクの中で最も優先順位
の低いタスクを検出
S24
そのタスクが利用する
レジスタセットIDを取得
S25
そのIDを指示する
終了

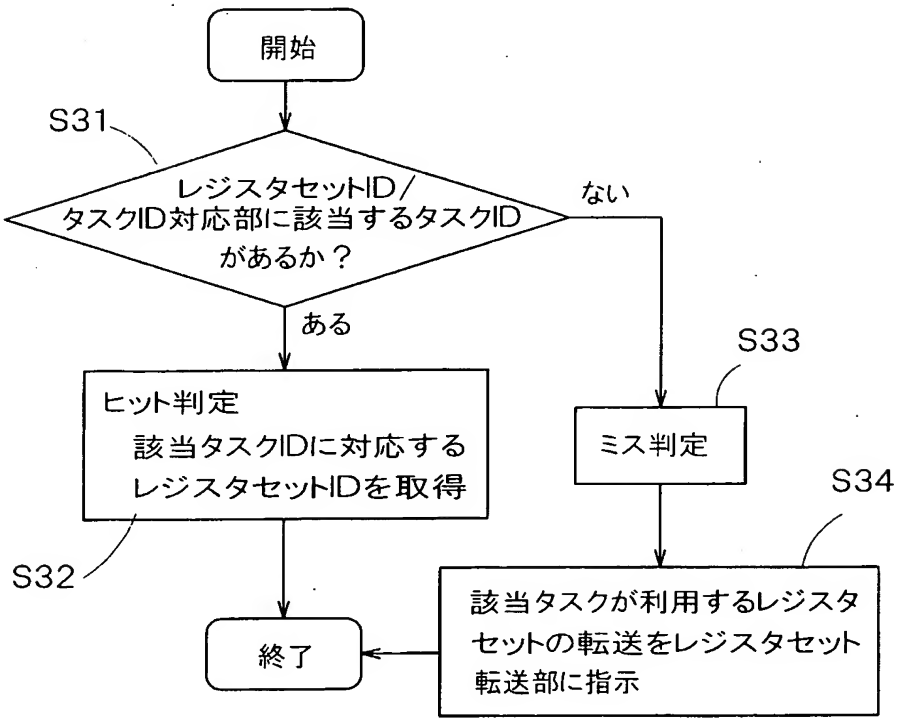
【図１７】


FIFO_0
FIFO_0
FIFO_0
10
FIFOステータス
タスク起動部
11
12
タスクID
14
転送終了
実行タスク決定部
優先順位
タスクID
タスクID・
優先順位
供給部
判定結果
31
タスクID レジスタセットID
判定要求
ヒット判定部
レジスタセットID
タスクID対応部
退避レジスタセット
決定部
24
35
レジスタセットID
レジスタセットイネーブル
21
レジスタセット1
R1
R0
SP
PC
22
レジスタ
セット
選択部
レジスタセット2
23
レジスタ
データ
演算部
レジスタセットm
演算結果
34
レジスタセット
転送部
1
2
n
PC
SP
R0
R1
33
外部記憶
制御部
32
外部レジスタセット記憶部

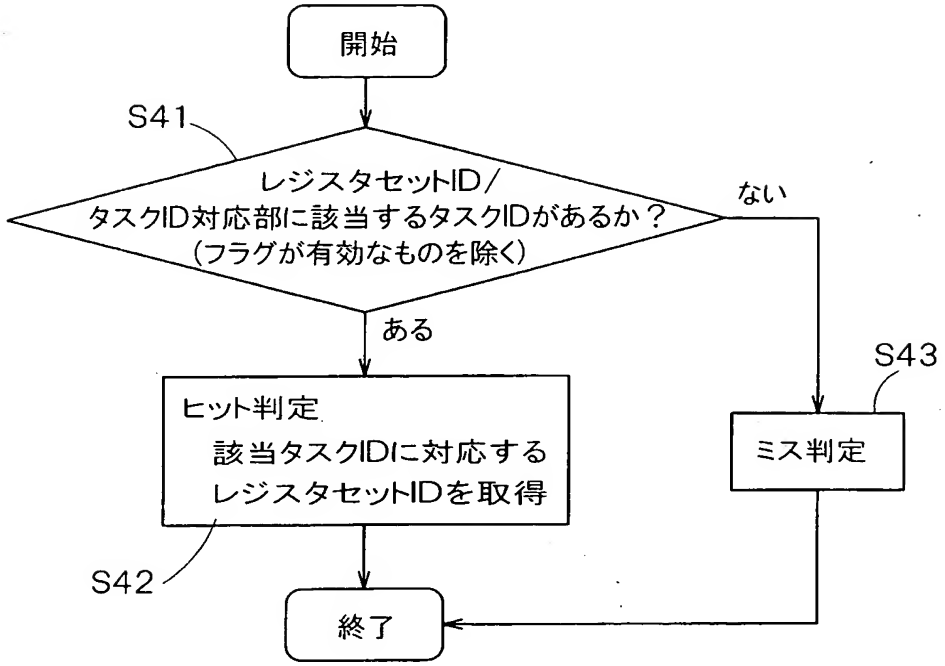
【図１８】


24
1
5
2
7
3
10
✓
4
1
m
2
レジスタセットID
タスクID
フラグ

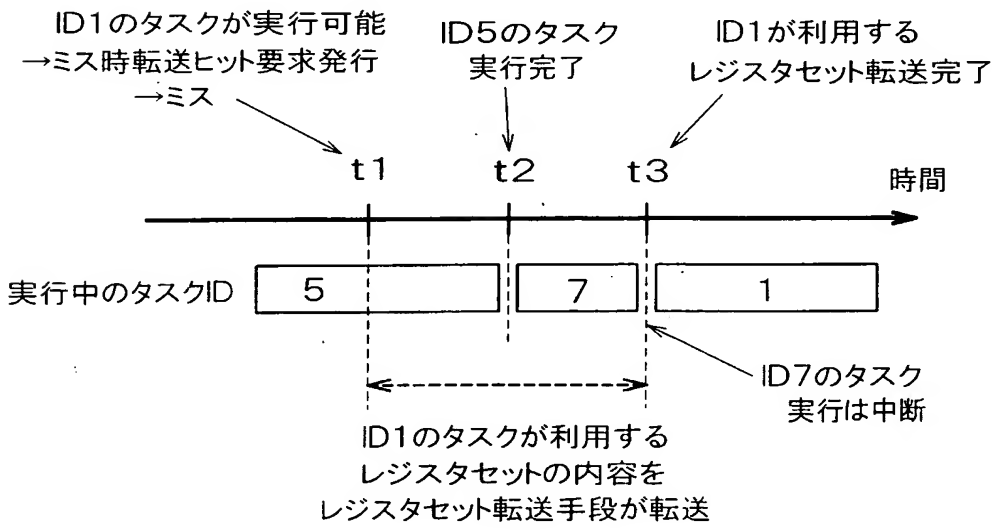
【図１９】


開始
S31
レジスタセットID/
タスクID対応部に該当するタスクID
があるか？
ない
ある
S33
ヒット判定
該当タスクIDに対応する
レジスタセットIDを取得
ミス判定
S34
S32
該当タスクが利用するレジスタ
セットの転送をレジスタセット
転送部に指示
終了

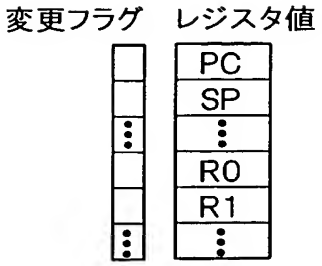
【図２０】


開始
S41
レジスタセットID/
タスクID対応部に該当するタスクIDがあるか？
（フラグが有効なものを除く）
ない
ある
S43
ヒット判定
該当タスクIDに対応する
レジスタセットIDを取得
ミス判定
S42
終了

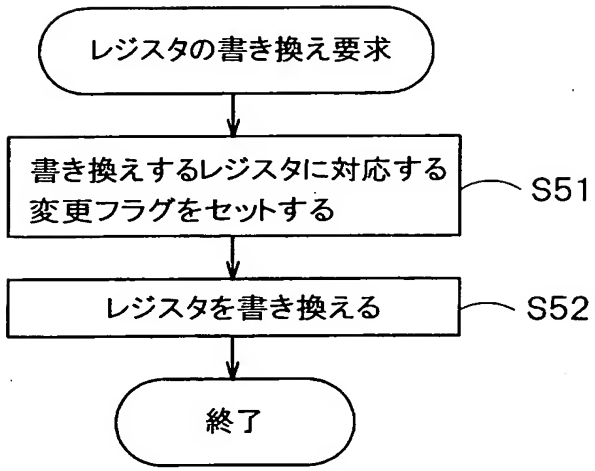
【図２１】


ID1のタスクが実行可能
→ミス時転送ヒット要求発行
→ミス
ID5のタスク
実行完了
ID1が利用する
レジスタセット転送完了
t1
t2
t3
時間
実行中のタスクID
5
7
1
ID7のタスク
実行は中断
ID1のタスクが利用する
レジスタセットの内容を
レジスタセット転送手段が転送

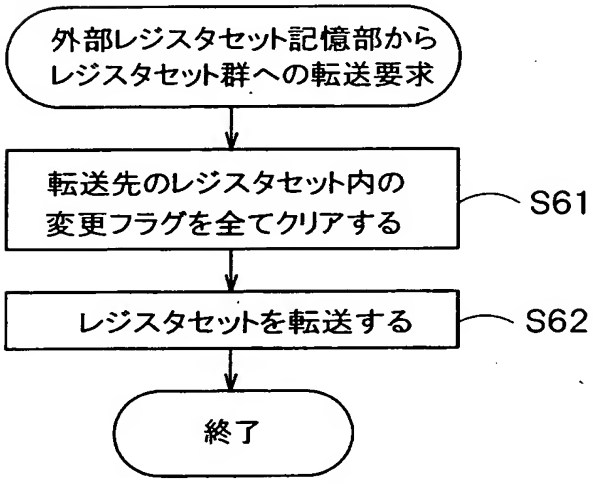
【図２２】


変更フラグ
レジスタ値
PC
SP
R0
R1

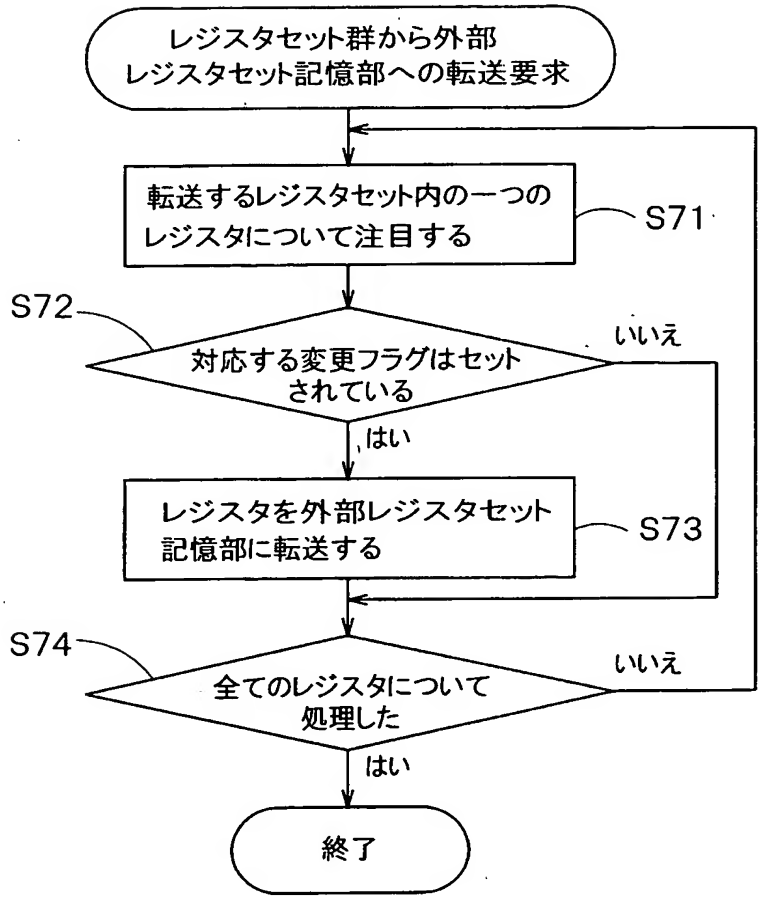
【図２３】


レジスタの書き換え要求
書き換えするレジスタに対応する
変更フラグをセットする
S51
レジスタを書き換える
S52
終了

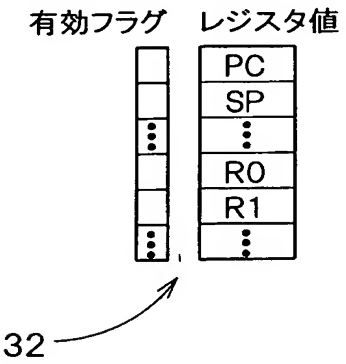
【図２４】


外部レジスタセット記憶部から
レジスタセット群への転送要求
転送先のレジスタセット内の
変更フラグを全てクリアする
S61
レジスタセットを転送する
S62
終了

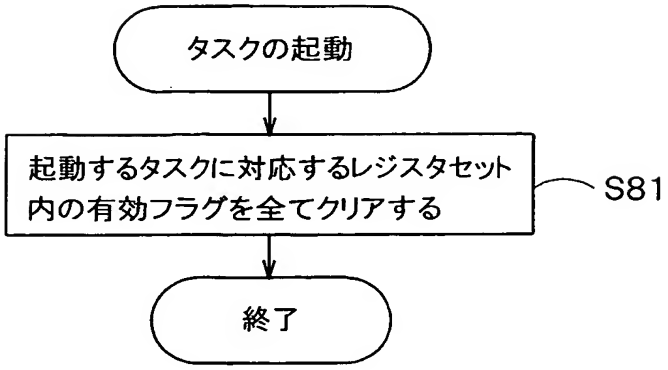
【図２５】


レジスタセット群から外部
レジスタセット記憶部への転送要求
転送するレジスタセット内の一つの
レジスタについて注目する
S71
S72
対応する変更フラグはセット
されている
いいえ
はい
レジスタを外部レジスタセット
記憶部に転送する
S73
S74
全てのレジスタについて
処理した
いいえ
はい
終了

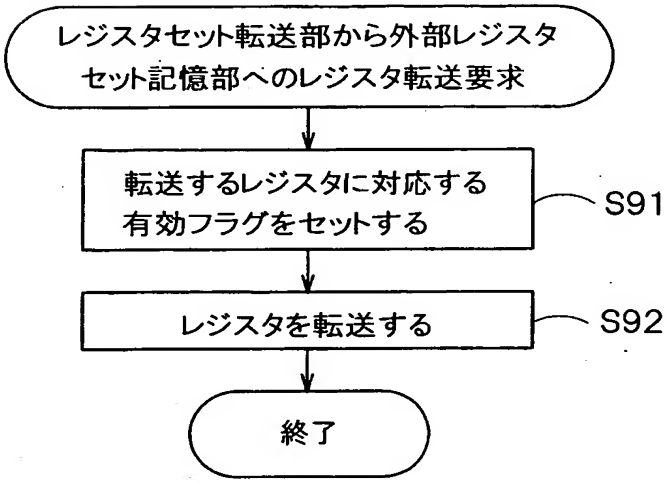
【図２６】


有効フラグ
レジスタ値
PC
SP
R0
R1
32

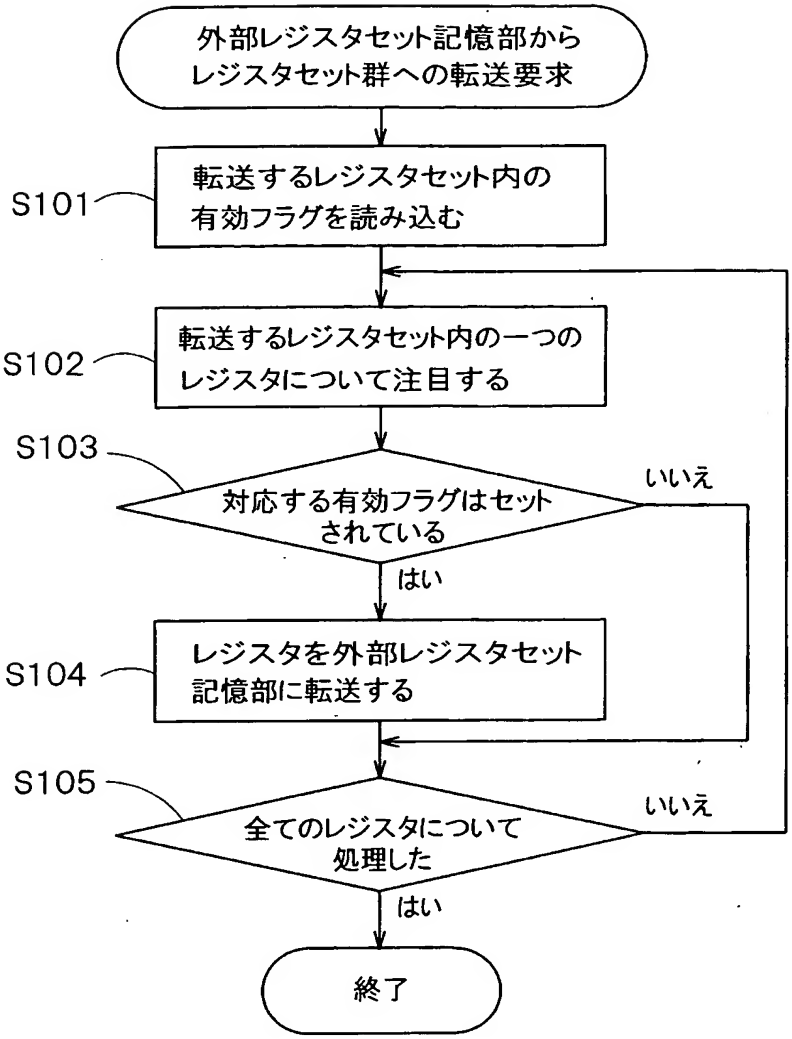
【図２７】


タスクの起動
起動するタスクに対応するレジスタセット内の有効フラグを全てクリアする
S81
終了

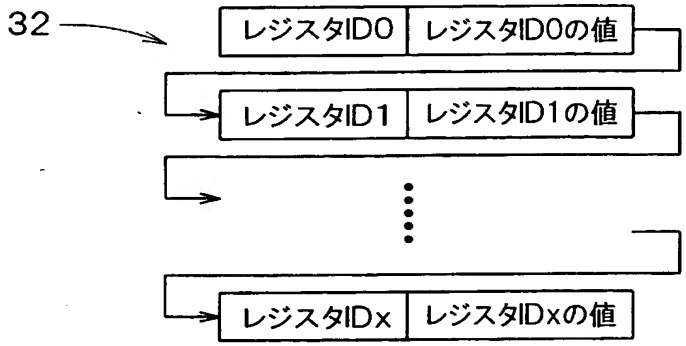
【図２８】


レジスタセット転送部から外部レジスタセット記憶部へのレジスタ転送要求
転送するレジスタに対応する有効フラグをセットする
S91
レジスタを転送する
S92
終了

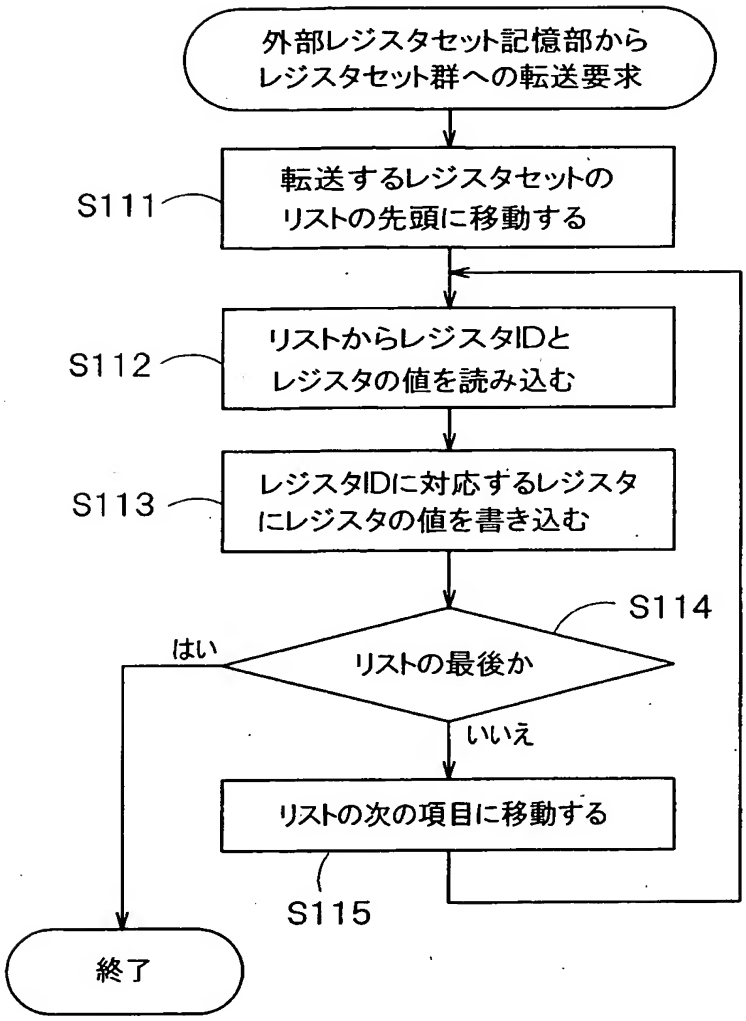
【図２９】


外部レジスタセット記憶部から
レジスタセット群への転送要求
S101
転送するレジスタセット内の
有効フラグを読み込む
S102
転送するレジスタセット内の一つの
レジスタについて注目する
S103
対応する有効フラグはセット
されている
いいえ
はい
S104
レジスタを外部レジスタセット
記憶部に転送する
S105
全てのレジスタについて
処理した
いいえ
はい
終了

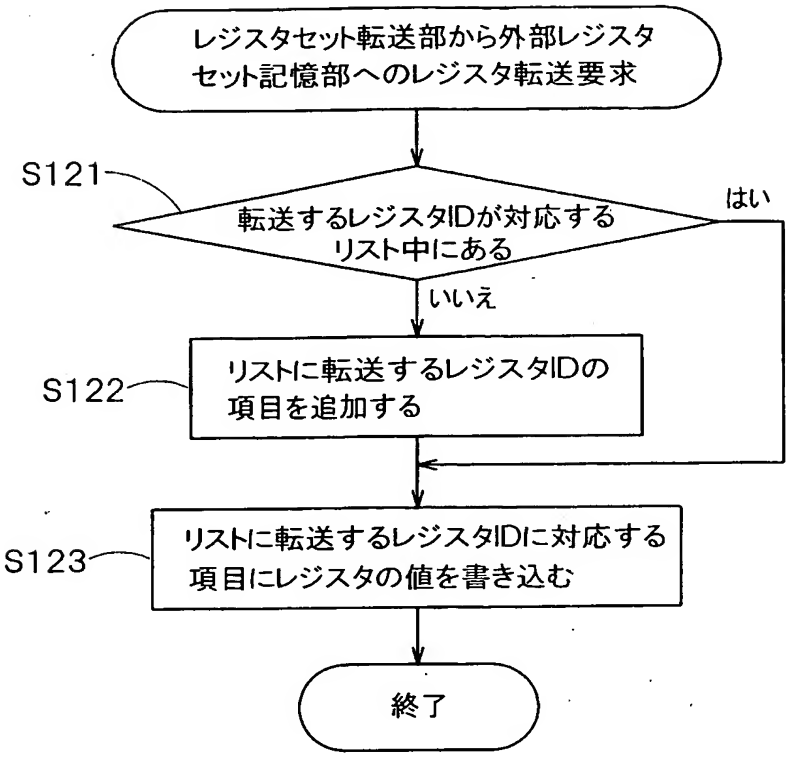
【図３０】


32
レジスタID0
レジスタID0の値
レジスタID1
レジスタID1の値
レジスタIDx
レジスタIDxの値

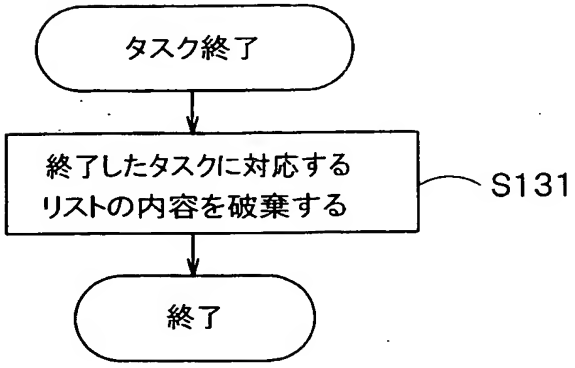
【図３１】


外部レジスタセット記憶部から
レジスタセット群への転送要求
S111
転送するレジスタセットの
リストの先頭に移動する
S112
リストからレジスタIDと
レジスタの値を読み込む
S113
レジスタIDに対応するレジスタ
にレジスタの値を書き込む
S114
はい
リストの最後か
いいえ
リストの次の項目に移動する
S115
終了

【図３２】


レジスタセット転送部から外部レジスタセット記憶部へのレジスタ転送要求
S121
転送するレジスタIDが対応するリスト中にある
はい
いいえ
S122
リストに転送するレジスタIDの項目を追加する
S123
リストに転送するレジスタIDに対応する項目にレジスタの値を書き込む
終了

【図３３】


タスク終了
終了したタスクに対応するリストの内容を破棄する
S131
終了

【書類名】　要約書

【要約】

【課題】　複数のタスクをリアルタイムに効率よく実行できるようにする。

【解決手段】　本発明に係るプロセッサは、複数のタスクそれぞれが起動可能か否かを判断するタスク起動部１１と、タスク起動部１１の判断結果に基づいて起動すべきタスクを決定する実行タスク決定部１２と、実行タスク決定部１２で決定されたタスクを実行するプロセッサコア１３とを備えている。タスク起動部１１には、複数のFIFO１０が接続されている。タスクに入力されるべきデータをFIFO１０が保持し、かつタスクの実行結果を格納すべき他のFIFO１０に空き領域が存在するか否かをタスク起動部１１で判断し、その判断結果に基づいて実行タスク決定部１２が起動すべきタスクを決定するようにしたため、オペレーティングシステムなしでタスクのスケジューリングを行うことができ、オペレーティングシステムのオーバーヘッドがなくなる。

【選択図】　図３

特願２００３－００３４２８

出　願　人　履　歴　情　報

識別番号　　［０００００３０７８］

１．変更年月日　　２００１年　７月　２日
［変更理由］　　住所変更
住　所　　東京都港区芝浦一丁目１番１号
氏　名　　株式会社東芝

２．変更年月日　　２００３年　５月　９日
［変更理由］　　名称変更
住所変更
住　所　　東京都港区芝浦一丁目１番１号
氏　名　　株式会社東芝